Universidad Politécnica de Madrid

Escuela Universitaria de Ingeniería Técnica de Telecomunicación

PROYECTO FIN DE CARRERA

Aplicaciones de filtrado adaptativo LMS para mejorar la respuesta de acelerómetros

CÉSAR E. WATANABE RUIZ

Septiembre 2012

Resumen

El proyecto, “Aplicaciones de filtrado adaptativo LMS para mejorar la respuesta de acelerómetros”, se realizó con el objetivo de eliminar señales no deseadas de la señal de información procedentes de los acelerómetros para aplicaciones automovilísticas, mediante los algoritmos de los filtros adaptativos LMS.

Dicho proyecto, está comprendido en tres áreas para su realización y ejecución, los cuales fueron ejecutados desde el inicio hasta el último día de trabajo.

En la primera área de aplicación, diseñamos filtros paso bajo, paso alto, paso banda y paso banda eliminada, en lo que son los filtros de butterworth, filtros Chebyshev, de tipo uno como de tipo dos y filtros elípticos.

Con esta primera parte, lo que se quiere es conocer, o en nuestro caso, recordar el entorno de Matlab, en sus distintas ecuaciones prediseñadas que nos ofrece el mencionado entorno, como también nos permite conocer un poco las características de estos filtros. Para posteriormente probar dichos filtros en el DSP.

En la segunda etapa, y tras recordar un poco el entorno de Matlab, nos centramos en la elaboración y/o diseño de nuestro filtro adaptativo LMS; experimentado primero con Matlab, para como ya se dijo, entender y comprender el comportamiento del mismo. Cuando ya teníamos claro esta parte, procedimos a “cargar” el código en el DSP, compilarlo y depurarlo, realizando estas últimas acciones gracias al Visual DSP.

Resaltaremos que durante esta segunda etapa se empezó a excitar las entradas del sistema, con señales provenientes del Cool Edit Pro, y además para saber cómo se comportaba el filtro adaptativo LMS, se utilizó señales provenientes de un generador de funciones, para obtener de esta manera un desfase entre las dos señales de entrada; aunque también se utilizó el propio Cool Edit Pro para obtener señales desfasadas, pero debido que la fase tres no podíamos usar el mencionado software, realizamos pruebas con el generador de funciones.

Finalmente, en la tercera etapa, y tras comprobar el funcionamiento deseado de nuestro filtro adaptativo DSP con señales de entrada simuladas, pasamos a un laboratorio, en donde se utilizó señales provenientes del acelerómetro 4000A, y por supuesto, del generador de funciones; el cual sirvió para la formación de nuestra señal de referencia, que permitirá la eliminación de una de las frecuencias que se emitirá del acelerómetro.

Por último, cabe resaltar que pudimos obtener un comportamiento del filtro adaptativo LMS adecuado, y como se esperaba. Realizamos pruebas, con señales de entrada desfasadas, y obtuvimos curiosas respuestas a la salida del sistema, como son que la frecuencia a eliminar, mientras más desfasado estén estas señales, mas se notaba. Solucionando este punto al aumentar el orden del filtro.

Finalmente podemos concluir que pese a que los filtros digitales probados en la primera etapa son útiles, para tener una respuesta lo más ideal posible hay que tener en cuenta el orden del filtro, el cual debe ser muy alto para que las frecuencias próximas a la frecuencia de corte, no se atenúen. En cambio, en los filtros adaptativos LMS, si queremos por

ejemplo, eliminar una señal de entre tres señales, sólo basta con introducir la frecuencia a eliminar, por una de las entradas del filtro, en concreto la señal de referencia. De esta manera, podemos eliminar una señal de entre estas tres, de manera que las otras dos, no se vean afectadas por el procedimiento.

Abstract

The project, "LMS adaptive filtering applications to improve the response of accelerometers" was conducted in order to remove unwanted signals from the information signal from the accelerometers for automotive applications using algorithms LMS adaptive filters.

The project is comprised of three areas for implementation and execution, which were executed from the beginning until the last day.

In the first area of application, we design low pass filters, high pass, band pass and band-stop, as the filters are Butterworth, Chebyshev filters, type one and type two and elliptic filters.

In this first part, what we want is to know, or in our case, remember the Matlab environment, art in its various equations offered by the mentioned environment, as well as allows us to understand some of the characteristics of these filters. To further test these filters in the DSP.

In the second stage, and recalling some Matlab environment, we focus on the development and design of our LMS adaptive filter; experimented first with Matlab, for as noted above, understand the behavior of the same. When it was clear this part, proceeded to "load" the code in the DSP, compile and debug, making these latest actions by the Visual DSP.

Will highlight that during this second stage began to excite the system inputs, with signals from the Cool Edit Pro, and also for how he behaved the LMS adaptive filter was used signals from a function generator, to thereby obtain a gap between the two input signals, but also used Cool Edit Pro himself for phase signals, but due to phase three could not use such software, we test the function generator.

Finally, in the third stage, and after checking the desired performance of our DSP adaptive filter with simulated input signals, we went to a laboratory, where we used signals from the accelerometer 4000A, and of course, the function generator, which was used for the formation of our reference signal, enabling the elimination of one of the frequencies to be emitted from the accelerometer.

Note that they were able to obtain a behavior of the LMS adaptive filter suitable as expected. We test with outdated input signals, and got curious response to the output of the system, such as the frequency to remove, the more outdated are these signs, but noticeable. Solving this point with increasing the filter order.

We can conclude that although proven digital filters in the first stage are useful, to have a perfect answer as possible must be taken into account the order of the filter, which should be very high for frequencies near the frequency cutting, not weakened. In contrast, in the LMS adaptive filters if we for example, remove a signal from among three signals, only enough to eliminate the frequency input on one of the inputs of the filter, namely the reference signal. Thus, we can remove a signal between these three, so that the other two, not affected by the procedure.

Agradecimientos

Quiero expresar mis más sinceros agradecimientos a mi familia, por su apoyo, ayuda, instrucción, paciencia y comprensión a lo largo de estos años universitarios cursados; y en especial un cordial y caluroso agradecimiento a mi madre.

A mis amigos, tanto fuera como dentro de la escuela, por aquellos momentos anecdóticos realizados dentro como también fuera de las instalaciones de la escuela; por su paciencia, tolerancia y comprensión.

Un cordial agradecimiento a mi tutor, Wilmar Hernández, por brindarme la oportunidad de realizar este proyecto expuesto en posteriores páginas, por su ayuda y motivación durante la realización de la misma.

Y sobre todo, agradecer a Dios por brindarme esta oportunidad de realizar una carrera universitaria.

A todos, **MUCHAS GRACIAS**.

INDICE

1. **INTRODUCCIÓN**

El presente proyecto se realizó con el objetivo de eliminar señales no deseadas de la señal de información procedente de los acelerómetros para aplicaciones automovilísticas, mediante los algoritmos de los filtros adaptativos LMS.

Se realizará una breve introducción a los filtros IIR y FIR, repasando un poco sus características, como también a sus distintos tipos y/o clases; comenzando por los filtros paso bajo, paso alto, paso banda, paso banda eliminada; y terminando en los filtros adaptativos LMS.

Para la realización y verificación de estos filtros y llegar a nuestro objetivo del proyecto, vamos a necesitar ciertas herramientas y utilidades, como son MATLAB, Cool Edit Pro, Visual DSP++ y el kit de desarrollo ADSP-21161 EZ-KIT.

Dichas herramientas y utilidades se usaron de la siguiente manera: MATLAB, nos permitió usar su interfaz para poder simular el comportamiento que podrían tener nuestros filtros digitales frente a diferentes excitaciones, permitiéndonos de esta manera observar en un plano virtual, como serán o podrían llegar a ser las respuestas a este “buffet” de entradas.

Con los filtros digitales simulados por MATLAB; pasamos a introducir, dichos filtros al DSP, con ayuda del Visual DSP++ y nuestro kit de desarrollo ADSP-21161 EZ-KIT. De esta manera “volcamos” nuestro pequeño código en el DSP y excitamos las entradas, con señales generadas por el Cool Edit Pro++. Así pudimos verificar el funcionamiento y comportamiento de nuestros filtros digitales frente a señales generadas a criterio nuestro por una herramienta software antes ya descrita.

Finalmente con estos datos de prácticas realizadas, nos vimos capaces de excitar el DSP, cargado ya con nuestro algoritmo del filtro digital, con señales producidas por un acelerómetro. Observando así, como responde nuestro algoritmo LMS a las señales que utilizamos o se utilizarán posteriormente para ejecución de nuestro proyecto.

Nos introduciremos un poco más en las herramientas y utilidades citadas, para entender un poco su funcionamiento y como se usó cada una de ellas.

Citaremos además, la variedad de uso o aplicaciones, que tienen los filtros adaptativos en general, en sus distintos tipos o clases; y los filtros adaptativos LMS, como son por ejemplo en aplicación para la reducción de ruido, estimación de canal, cancelaciones de errores,

identificaciones de sistemas, eliminación de interferencias, etc. Nombraremos a su vez, conforme expliquemos un poco dichas aplicaciones, algunos de los muchos artículos donde pudimos encontrar esta información.

Para acabar con esta introducción destacaremos nuevamente que el objetivo principal del presente proyecto, se dedicó a la eliminación de señales no deseadas de señales útiles mediante filtros adaptativos LMS para aplicaciones automovilísticas.

2. NOCIONES TEÓRICAS DE FILTRADO DIGITAL DE LA SEÑAL

Un filtro es un sistema (Figura 1) que, dependiendo de algunos parámetros, realiza un proceso de discriminación de una señal de entrada obteniendo variaciones en su salida.

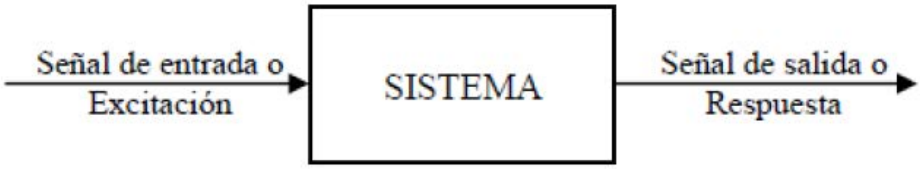

Figura 1.Diagrama de bloques de un sistema.

Los filtros digitales tienen como entrada una señal digital y a su salida tienen otra señal digital (Figura 2), pudiendo haber cambiado en amplitud, frecuencia o fase dependiendo de las características del filtro.

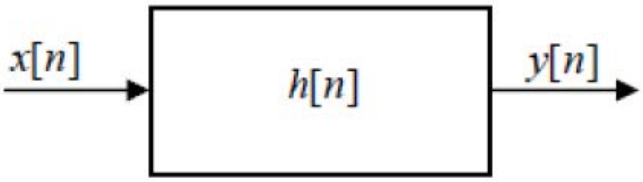

Figura 2. Diagrama de bloques de un sistema en tiempo discreto

La ecuación en diferencia de un sistema es:

$$\sum_{m=0}^{N_a-1} a_m y[n-m] = \sum_{k=0}^{N_b-1} b_k x[n-k] \quad (1)$$

Aplicando la propiedad de superposición de los sistemas LTI (Lineales e invariantes en el tiempo), se puede determinar la salida del sistema ante una cierta entrada de la manera siguiente:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k]$$

(2)

Siendo h[n] la respuesta o salida del sistema ante una entrada equivalente a un impulso unitario δ[n], denominada *respuesta al impulso del sistema*. El segundo miembro de la expresión representa el producto de convolución de la señal de entrada x[n] y la respuesta al impulso del sistema h[n], esto es:

$$Y[n] = x[n] * h[n] = h[n] * x[n]$$

(3)

En el caso aplicar la transformada Z a la ecuación en diferencias de (1), se obtiene la función de transferencia del sistema:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{N_a-1} a_m y[n-m]}{\sum_{k=0}^{N_b-1} b_k x[n-k]}$$

(4)

Básicamente, un filtro es un dispositivo que impide o permite el paso de una cierta gama de frecuencias, donde permitir o impedir está relacionado con un nivel de atenuación o ganancia. También sirven para restaurar una señal, cuando haya una señal que haya sido deformado de alguna forma. La separación de señales es necesaria cuando una señal ha sido contaminada con interferencias, ruidos u otras señales.

El filtrado digital consiste en la realización de un procesado de datos de entrada. El valor de la muestra de la entrada actual y algunas muestras anteriores (que previamente habían sido almacenadas), son multiplicados por unos coeficientes definidos. También podría tomar valores de la salida en instantes pasados y multiplicarlos por otros coeficientes. Finalmente todos los resultados de todas estas multiplicaciones son sumados, dando una salida para el instante actual. Esto implica que internamente tanto la salida como la entrada

del filtro serán digitales, por lo que puede ser necesaria una conversión analógico-digital o digital-analógica para uso de filtros digitales en señales analógicas.

Al ser el filtro digital un sistema de tiempo discreto que puede realizar funciones de filtrado de señales, éste debe cumplir los requisitos necesarios para procesar las señales analógicas (Teorema del muestreo o Nyquist-Shannon).

La parte analógica de la señal debe ser previamente muestreada y digitalizada por un convertidor AD (analógico-digital). Los números binarios resultantes de la conversión anterior, que representan valores sucesivos muestreados de la señal de entrada, son transferidos al procesador (en nuestro caso al DSP), que realiza los cálculos numéricos explicados anteriormente (multiplicaciones y sumas). Si es necesario, los resultados de los cálculos, que representan los valores de una señal filtrada, son encaminados a través de un convertidor DA (Digital-Analógico) para convertir la señal a su forma analógica.

En la siguiente figura se muestra la configuración básica de un filtrado digital:

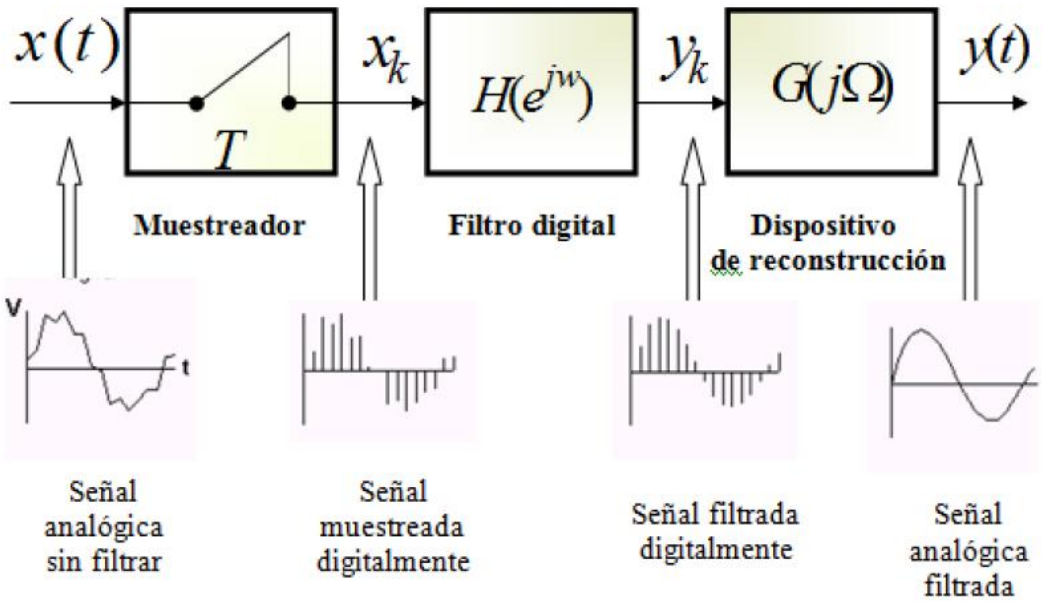

Figura 3. Estructura filtrado digital

Que con más detalle queda de la siguiente forma:

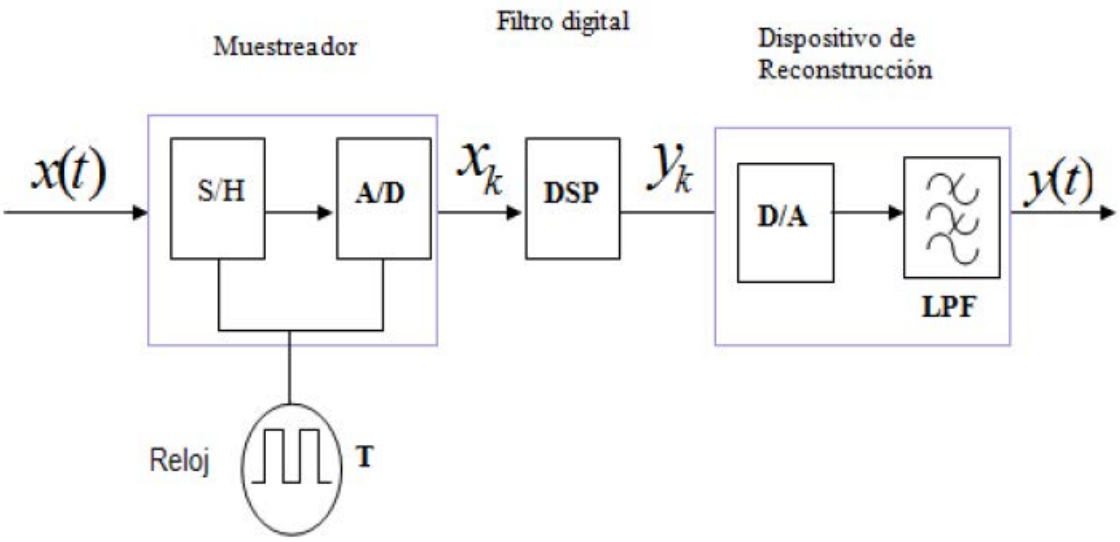

Figura 4. Estructura detallada del filtrado digital

2.1 **TIPOS DE FILTROS**

Hay varios tipos de filtros, así como distintas clasificaciones para estos filtros. De acuerdo con la parte del espectro que dejan pasar y que atenúan hay:

- Filtros paso alto.
- Filtros paso bajo.
- Filtros paso banda.
- Filtros paso banda eliminada.
- Filtros paso todo.
- Etc.

De acuerdo con su orden:

- Primer orden.
- Segundo orden, etc.

De acuerdo con el tipo de respuesta ante entrada unitaria:

- FIR (Finite Impulse Response)
- IIR (Infinite Impulse Response)

2.1.1 FILTROS DIGITALES FIR (FILTROS NO-RECURSIVOS).

Los filtros digitales de Respuesta Finita al impulso o filtros FIR (por sus siglas en ingles, Finite Impulse Response), se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número finito de términos no nulos. La estructura de la señal a la salida del filtro se basa solamente en la combinación lineal de las entradas actuales y anteriores (finaliza cuando la señal de entrada lo hace).

Las características más relevantes de este tipo de filtros son:

- Tienen respuesta al impulso finita.
- No tienen realimentación.
- No tienen problemas de estabilidad ya que todos sus polos están en el origen.

La estructura de señal a la salida de este tipo de filtro se basa solamente en la combinación lineal de las entradas actuales y anteriores, esto es:

$$H(z) = \sum_{k=0}^{N} a_k . z^{-k} \tag{5}$$

En la figura 5, se muestra el diagrama de bloques de la estructura básica del filtro FIR, para una cantidad de 12 coeficientes.

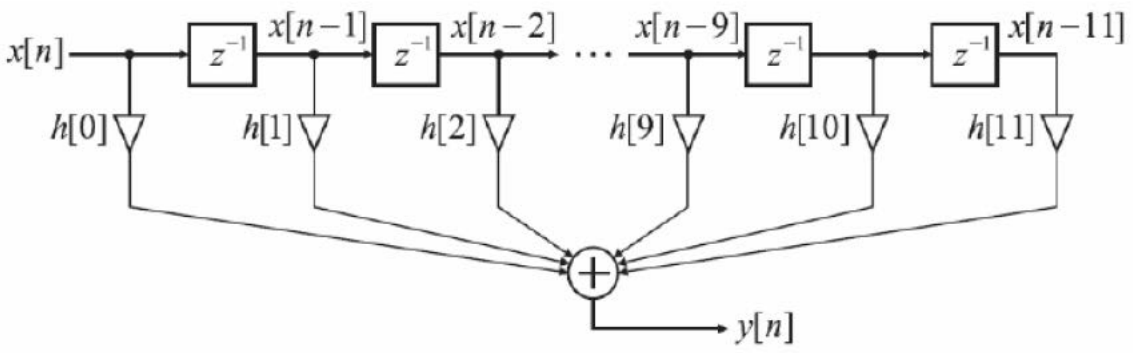

Figura 5. Diagrama de bloques de un filtro FIR

En la siguiente figura se muestran los módulos de las respuestas en frecuencia de los filtros más comunes:

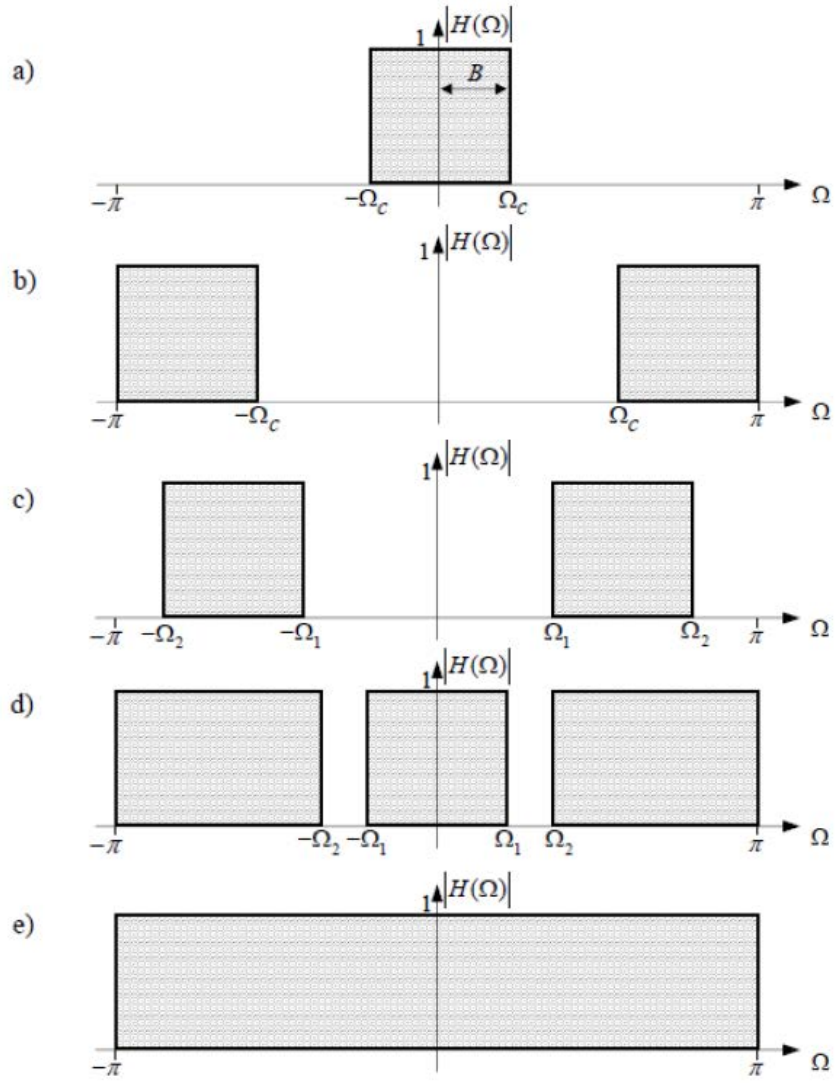

Figura 6. Respuesta en frecuencia de los diferentes tipos de filtros:

a) FPB, b) FPA, c) FPBanda, d) FPBanda Eliminada e) FPTodo

En la práctica, las respuestas en frecuencia mostradas en la Figura 6, no se obtienen.

Un filtro implementado físicamente tiene bandas de paso, transición y rechazo y no sólo frecuencias de corte. Un filtro paso bajo cuya respuesta en frecuencia tiene estas características, se muestra en la figura 7.

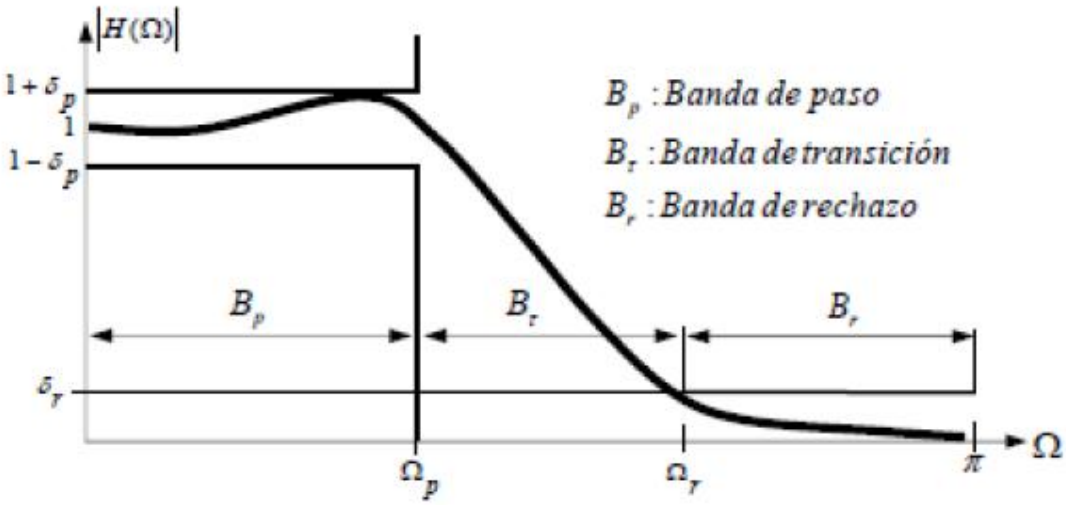

Figura 7. Respuesta real de un filtro

En el diseño de un filtro analógico, la frecuencia de corte se encuentra habitualmente donde la amplitud se reduce a 0,707, sin embargo, en los filtros digitales se encuentra menos estandarizado, y es común encontrarse con el 99%, 90%, 70,7% y 50% de los niveles de amplitud definidos para la frecuencia de corte.

2.1.2 FILTROS DIGITALES IIR (FILTROS RECURSIVOS)

Los filtros digitales de Respuesta Infinita al Impulso o filtros IIR (por sus siglas en inglés, Infinite Impulse Response), se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso la salida tendrá un número infinito de términos no nulos.

Un filtro recursivo es aquel que añade a los valores de entrada algún valor de salida previo. Estos, al igual que las entradas, son almacenados en la memoria del procesador.

Las características más relevantes de este tipo de filtros son:

- Tienen respuesta al impulso de duración infinita.
- Tienen realimentación.
- Deben diseñarse con cuidado para evitar problemas de estabilidad.

La ecuación que caracteriza a los filtros IIR es:

$$H(z) = \frac{\sum_{k=0}^{M} b_k . z^{-k}}{1 + \sum_{k=1}^{N} a_k . z^{-k}} \tag{6}$$

A continuación se puede observar el funcionamiento de un filtro recursivo a través de un diagrama de bloques:

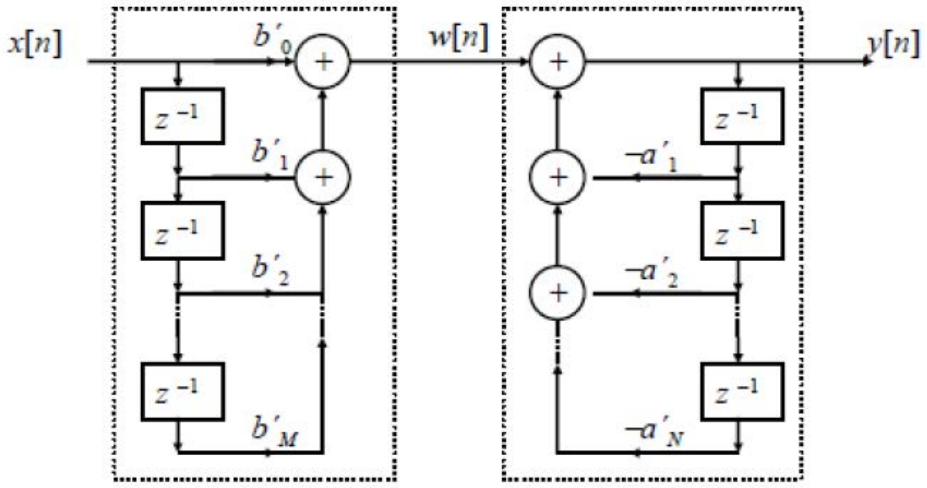

Figura 8. Diagrama de bloques de un filtro IIR

Que simplificado para ahorrar en número de retardos queda:

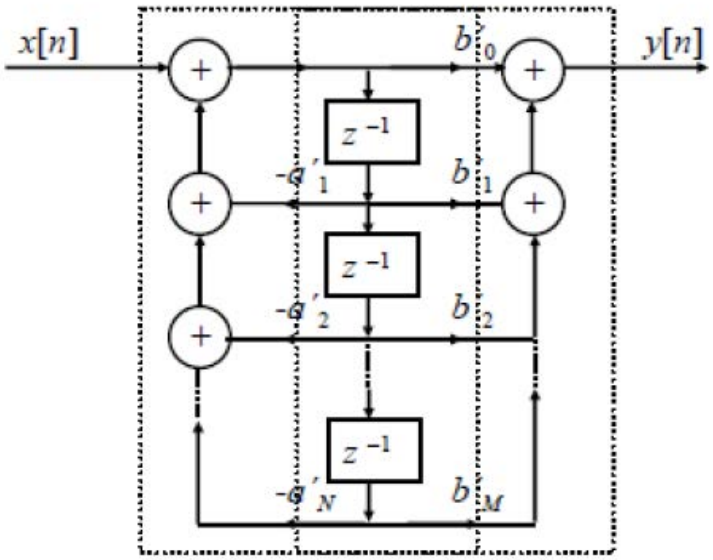

Figura 9. Diagrama de bloques de un filtro IIR simplificado

Se puede pensar que en este tipo de filtros se requieren más cálculos para ejecutar el filtrado de la señal, pues en una parte de los valores de entrada, también se encuentran términos de salida en la expresión del filtro. De hecho pasa todo lo contrario, ya que un filtro recursivo requiere una expresión de menor orden (y por tanto, muchos menos términos a tratar por el procesador) que su equivalente filtro no-recursivo.

2.2. FILTRO IIR BUTTERWORTH

Se caracteriza por producir la respuesta más plana posible en la banda de paso, es decir no presenta rizado, donde decaerá linealmente desde la frecuencia de corte hacia el infinito, a razón de 20n dB por década ó 6n dB por octava (siendo n el número de polos del filtro).

Cabe resaltar, que a medida que aumentamos el orden del filtro la respuesta es más ideal, es decir mientras mas aumentemos el orden, la fase es menos lineal.

2.3. FILTRO IIR CHEBYSHEV

Dependiendo del tipo del filtro, presentará o bien un rizado en la banda de paso, tipo uno, o en la banda atenuada, tipo dos.

Es decir Chebyshev tipo uno maximiza la pendiente fuera de la banda de paso, presenta rizado en la banda de paso y es monolítico fuera de ella.

En cambio Chebyshev tipo dos presenta una respuesta monolítica en la banda de paso y rizado en la banda de rechazo.

A diferencia de Butterworth, la respuesta transitoria es peor.

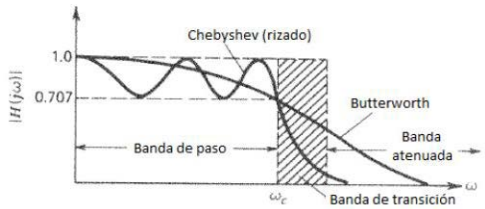

Figura 10. Comparación de respuesta entre filtro Butterworth y filtro Chebyshev tipo 1

2.4. FILTRO ELÍPTICO O DE CAUER

Esta clase de filtros se caracterizan por tener rizado en la banda de paso y en la banda atenuada, utilizados para eliminación de una frecuencia en concreto.

Suelen ser más eficientes debido a que al minimizar la zona de transición, ante unas mismas restricciones, consiguen un menor orden.

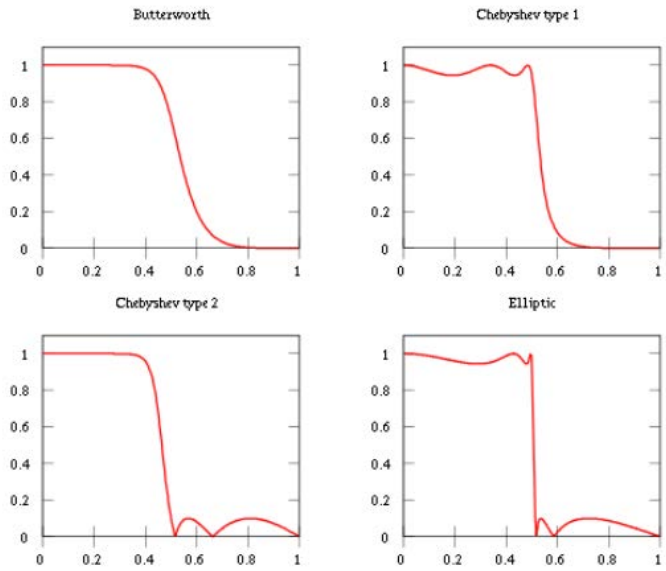

Figura 11. Diferencias en la respuesta de los filtros Butterworth, Chebyshev y Eliptico

2.5. DISEÑO DE FILTROS ADAPTATIVOS

Este tipo de filtros se caracterizan por modelar la relación entre señales en tiempo real de forma iterativa.

A diferencia de los filtros digitales IIR o FIR, los filtros adaptativos pueden cambiar su forma de comportarse, es decir pueden cambiar sus coeficientes de acuerdo con un algoritmo adaptativo. Por consiguiente no se saben los coeficientes del filtro cuando se diseña, dichos coeficientes se calculan cuando se implementa el filtro y se reajustan automáticamente en cada iteración.

Otra característica de estos filtros es que no son invariantes en el tiempo y tampoco son lineales, por lo que su estudio es más complejo que el de un filtro digital.

Normalmente estos tipos de filtros se implementan en forma de algoritmos sobre microprocesadores, DSP o FPGA.

El filtro adaptativo, es un sistema el cual presenta dos señales como entrada: x(n) y e(n); siendo esta última la señal de error, proveniente de la resta de una señal, llamada “señal deseada” (d(n)), y otra que es la señal de salida del filtro (y(n)). Además de estas señales, tenemos los coeficientes del filtro, llamados w(n), que son las que se multiplican a la entrada x(n) para obtener la salida.

La teoría de filtrado adaptativo es fundamental para el control inverso adaptativo. Los filtros adaptativos se usan para modelado, modelado inverso, y para hacer cancelaciones de ruido de la planta. Se va a presentar el filtro adaptativo como un bloque, que tiene una señal de entrada, una de salida, y una señal de entrada especial llamada “el error”, que se usa en el proceso de aprendizaje. El propósito de este apartado es ofrecer una introducción a la teoría de filtrado digital adaptativo.

El filtro adaptativo con el cual nos hemos basado durante el desarrollo de los filtros LMS y filtros predictivos:

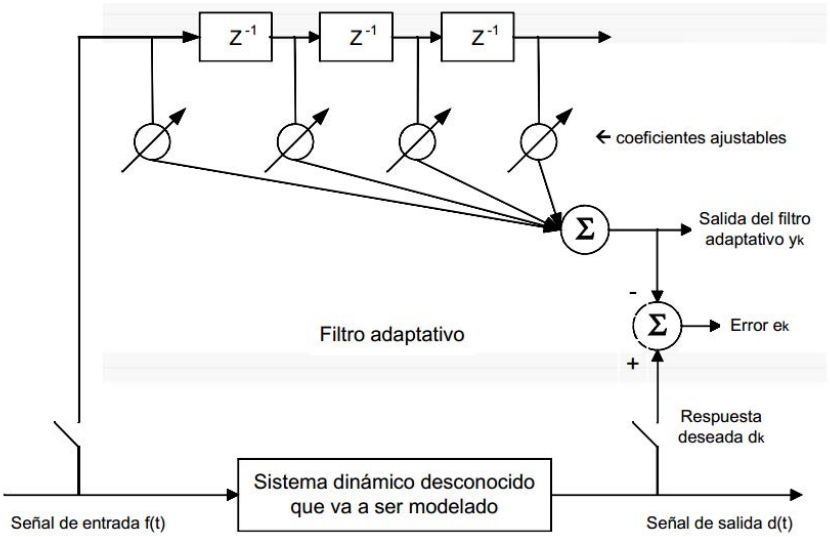

Figura 12. Modelado de un sistema mediante un filtro adaptativo

Además de las señales ya comentadas, hay que utilizar una más: la señal deseada, para poder generar la señal de error. La salida de la planta desconocida nos da el valor de la señal deseada.

El filtro adaptativo de la figura 11, es un filtro digital. El sistema desconocido que va a ser modelado es un filtro analógico. Las entradas al filtro adaptativo son, por lo tanto, versiones muestreadas de las señales de entrada y salida del sistema desconocido. Los coeficientes del filtro adaptativo se ajustan automáticamente mediante un algoritmo que minimiza el error cuadrático medio. Cuando los coeficientes convergen y el error se hace pequeño, la respuesta impulsiva del filtro adaptativo es muy parecida a la del sistema desconocido

El análisis del filtro adaptativo puede ser desarrollado considerando primero el combinador lineal adaptativo de la figura 12, un subsistema de la figura 11.

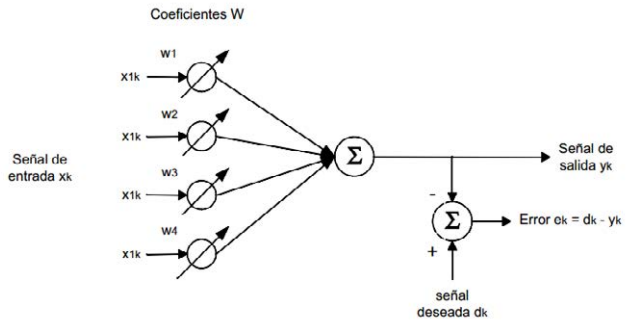

Figura 13. Combinador adaptativo lineal

Se tienen las siguientes señales:

- La señal de entrada, que es un conjunto de "n" señales. La k-ésima señal de entrada es:

$$X_k = [x_{1k}, x_{2k}, \dots, x_{lk}, \dots, x_{nk}]^T \tag{7}$$

- El conjunto de coeficientes o pesos se designan con el vector:

$$W^T = [w_1, w_2, \dots, w_l, \dots, w_n] \tag{8}$$

- La señal de salida k-ésima será:

$$y_k = \sum_{l=1}^{n} w_l x_{lk} = W^T X_k \tag{9}$$

Si se denota la señal deseada como $\boldsymbol{d_k}$ el error en el instante k-ésimo es:

$$e_k = d_k - y_k = d_k - W^T X_k \tag{10}$$

El error cuadrático medio es el valor esperado de $\boldsymbol{e_k^2}$**:**

$$MSE = \mathrm{E}\left[e_k^2\right] = \mathrm{E}\left[d_k^2\right] - 2\mathrm{E}\left[d_k X_k^T\right]W + W^T\mathrm{E}\left[X_k X_k^T\right]W = \mathrm{E}\left[d_k^2\right] - 2P^T W + W^T RV \tag{11}$$

Siendo **P** la correlación cruzada entre la señal de entrada **x** y la señal deseada **d**, y **R** la autocorrelación de la señal de entrada $\mathbf{E}\left[\boldsymbol{X_k X_k^T}\right]$**.** Se puede observar que el MSE es una función cuadrática de los coeficientes, con lo que tendrá la forma de una parábola. El proceso adaptativo estará continuamente ajustando los coeficientes, buscando la parte más baja de la parábola. Para obtener los coeficientes óptimos, habrá que derivar el MSE e igualarlo a cero.

2.6. DISEÑO DE FILTRO LMS

Del inglés Least-Mean-Square; el algoritmo LMS utiliza el método ***steepest descent***[1] para la actualización de los coeficientes del vector de pesos. Este algoritmo realiza los cambios de forma proporcional al gradiente, tal como se puede ver en la siguiente ecuación:

[1] ***steepest descent,*** *es una extensión del método de Laplace para aproximar una integral.*

$$W_{k+1} = W_k + \mu(-\widehat{\nabla}_k) \tag{12}$$

Siendo ∇la estimación del gradiente:

$$\widehat{\nabla}_k = \nabla + N_k \tag{13}$$

Y N_k el ruido del gradiente.

El gradiente se define como la derivada del cuadrado del error respecto a cada uno de los coeficientes. Si se hace esa derivada, se obtiene como conclusión que el gradiente tiene el valor:

$$\widehat{\nabla}_k = -e_k X_k \tag{14}$$

Sustituyendo en la ecuación anterior, se obtiene la ecuación que rige al algoritmo LMS:

$$W_{k+1} = W_k + \mu e_k X_k \tag{15}$$

Con estas ecuaciones podremos definir un algoritmo, el cual tendrá la siguiente forma:

1. Inicializar los pesos
2. Elegir un valor para μ
3. Calcular la salida y(n)
4. Calcular el error e(n)
5. Actualizar los pesos con la función de coste elegida
6. Repetir un determinado número de veces desde el punto 3.

2.7. DISEÑO DE FILTROS PREDICTIVOS

Para el diseño de los filtros de predictivos, nos basamos en la lógica de los filtros adaptativos LMS, por lo que prácticamente realizamos un ALE, Adaptive Line Enhancer

Esta clase de filtros (filtros predictivos o ALE), se caracterizan por tener asociadas una señal de entrada, en el instante inicial, como señal deseada; y la misma señal de entrada pero retrasada como nuestra señal **"x"**.

Partiendo de estos datos, podríamos utilizar las ecuaciones antes expuestas, para posteriormente codificar el algoritmo.

En la siguiente figura, podremos observar cómo sería el diagrama de cuadros:

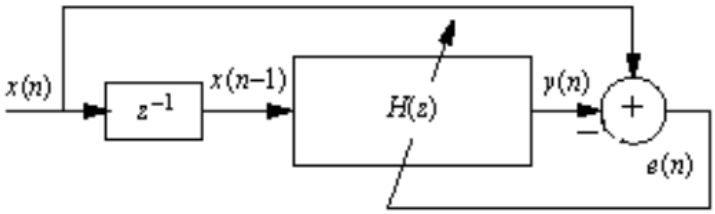

Figura 14. Diagrama de un filtro de predicción.

Como bien se puede observar en la figura 13, la señal deseada es la entrada instantes más adelante, siendo la señal **x**, la señal de entrada (señal deseada), retrasada un instante en el tiempo. Además H(z) ajustaría sus parámetros para minimizar el error.

También deberíamos suponer que la estructura de la serie temporal no cambia o en todo caso lo hace muy lentamente de modo que el sistema pueda adaptarse a los cambios.

Entonces, si modelamos un sistema desconocido M(z), que produce la señal x(n) ante una entrada de ruido blanco.

Una vez minimizado el error, la salida de todo el dispositivo será de nuevo ruido blanco. El bloque D(z) actúa como inverso del sistema desconocido, por lo que los polos de D serán los ceros de M y viceversa.

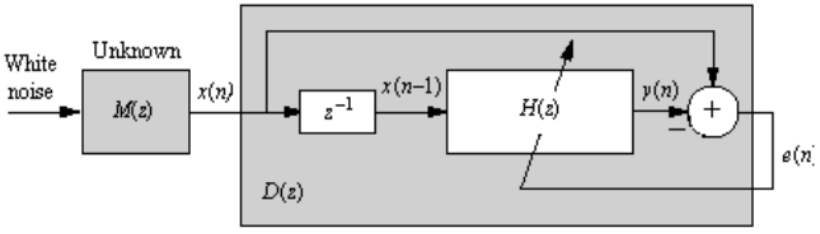

Figura 15. Diagrama de modelación del sistema M(z)

Es decir, la señal e(n), será ruido blanco, y nuestra señal y(n), representará la salida global del sistema, la cual tendrá la señal útil.

3. HERRAMIENTAS HARDWARE

En cuanto a la descripción hardware del entorno de desarrollo, cabe resaltar que la gestión y desarrollo del presente proyecto gira en torno a la idea de realizar programas para el filtrado de señales con el software Visual DSP++ 4.5 y cargar dichos programas en la tarjeta hardware conectada al PC. En nuestro caso, la placa sobre la que se cargarán, se ejecutarán y se observarán los resultados de filtrado es el "KIT de evaluación" ADSP 21161-N. Por ello, a continuación pasaremos a describir detenidamente el mencionado KIT desarrollo y sus prestaciones.

3.1. EVALUATION KIT ADSP 21161-N

Este "KIT de evaluación" está diseñado para ser utilizado conjuntamente con el entorno de desarrollo VisualDSP++ y hacer pruebas sobre el procesador ADSP-21161N. Dicho entorno ha sido descrito anteriormente con detalle habiendo observado las múltiples opciones que ofrece, como por ejemplo:

- Crear, compilar, cargar y ejecutar programas escritos en lenguaje C++.
- Crear proyectos.
- Visualización de variables.

A partir de la siguiente figura describiremos los recursos del "KIT de evaluación" que vamos a utilizar en el desarrollo del proyecto, así como la interconexión de los mismos con el resto del entorno de desarrollo:

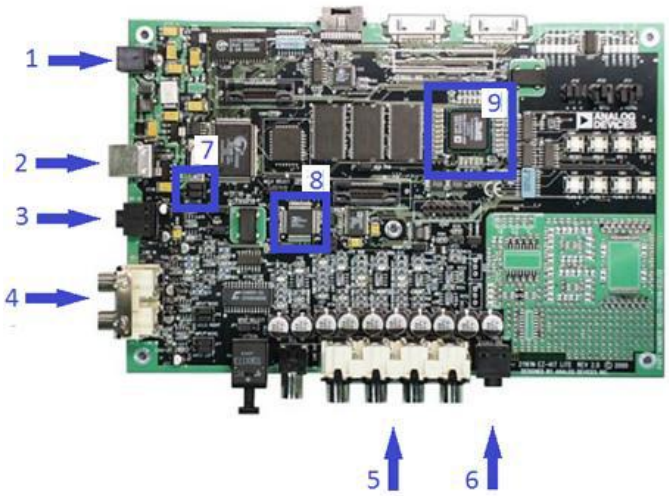

Figura 16. Vista de la tarjeta de desarrollo (ADSP 21161-N).

1. Conector de alimentación
2. Conector USB
3. Conector stereo jack de entrada
4. Conectores RCA de entrada
5. Conectores RCA de salida
6. Conector stereo jack de salida
7. Jumpers de selección de canal de entrada
8. Códec de audio Multicanal AD1836
9. ADSP 21161-N

Con el fin de crear una idea acerca del funcionamiento y la interacción de todos los recursos específicos, procederemos a comentar detalladamente el funcionamiento e interconexión de cada uno de ellos.

3.1.1. CONECTOR DE ALIMENTACIÓN

La placa se alimenta con una tensión continua de 7V, suministrada por la fuente de alimentación que viene incluida en el “Evaluation KIT” y que conecta la red de 220V con el conector.

3.1.2. CONECTOR USB

Para interconectar la placa con el ordenador y poder cargar los programas en el DSP, el "KIT de evaluación" viene provisto de un cable USB 2.0 para poder realizar la conexión entre ambos.

3.1.3. CONECTOR STEREO JACK DE ENTRADA

Este conector jack de 3.5mm es uno de los canales que puede elegirse como entrada de datos. Puede multiplexarse entre él o dos conectores RCA como canal de entrada, como explicaremos más adelante. Una vez elegido, la señal de entrada se dirige hacia el AD1836.

3.1.4. CONECTORES RCA DE ENTRADA

Son 4 y componen dos canales de entrada de datos. Cada canal posee a su vez dos canales (stereo): Canal Derecho (conector rojo) y Canal Izquierdo (conector blanco). Además, están denominados en la placa como Channel 1 y Channel 2.

El Canal 1 (Channel 1) tiene la opción de elegirse como entrada o dejarlo "al aire" y elegir en su lugar el jack stereo de entrada mediante una multiplexación basada en jumpers que explicaremos más adelante.

El canal 2 (Channel 2) no tiene la posibilidad de ser multiplexado.

Tanto el Canal 1 (si lo escogemos como entrada) como el Canal 2, dirigen su señal de entrada hacia el AD1836.

3.1.5. CONECTORES RCA DE SALIDA

Existen más que de entrada, son 8 y componen cuatro canales de salida provenientes de DAC's del AD1836. Al igual que los de entrada, los RCA de salida también componen canales "por parejas" y están nombrados en la placa como Channel 1, Channel 2, Channel 3 y Channel 4.

3.1.6. CONECTOR STEREO JACK DE SALIDA

Este conector es como el que se utiliza para la entrada pero usado como salida.

3.1.7. JUMPERS DE SELECCIÓN DE CANAL DE ENTRADA

Esta placa permite, mediante jumpers, a uno de los dos canales disponibles seleccionar "su entrada" por la que se realizará la adquisición de los datos a tratar en el DSP.

Como hemos mencionado antes, puede realizarse una multiplexación entre el jack stereo de entrada y los conectores RCA del Canal 1 de entrada. Una vez elegida cuál será la entrada, los datos se dirigen al AD1836. Concretamente, al ADC1.

En la siguiente figura se muestra la multiplexación que se puede hacer de esta entrada:

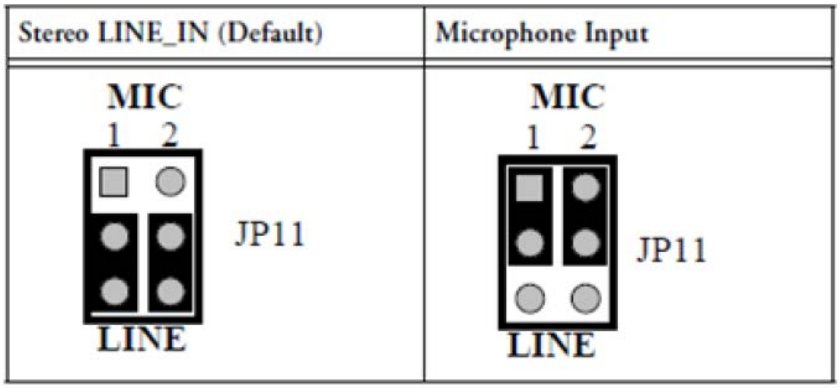

Figura 17. Conexiones del Jumper JP11

Si están conectados los pines 1-3 y 2-4 del Jumper JP11 (etiquetado en la placa), la entrada seleccionada para el canal es el jack stereo. Si por el contrario, los pines seleccionados son 3-5 y 4-6, la entrada corresponde a los RCA del Canal 1.

Una vez seleccionada la entrada, los datos son enviados al ADC1 del AD1836. Los otros dos RCA que componen el Canal 2 de entrada conectan con el ADC2 del AD1836 (más adelante se explicará los elementos que componen el AD1836).

3.1.8. PLATAFORMA ADSP-21161N EZ-KIT LITE: SISTEMA DE AUDIO DE DISEÑO AD1836/ADSP-21161.

La plataforma ADSP-21161N EZ-KIT Lite, incluye el DSP, ADSP-21161 SIMD SHARC, el códec AD1836, AD 1852 y SP/DIF.

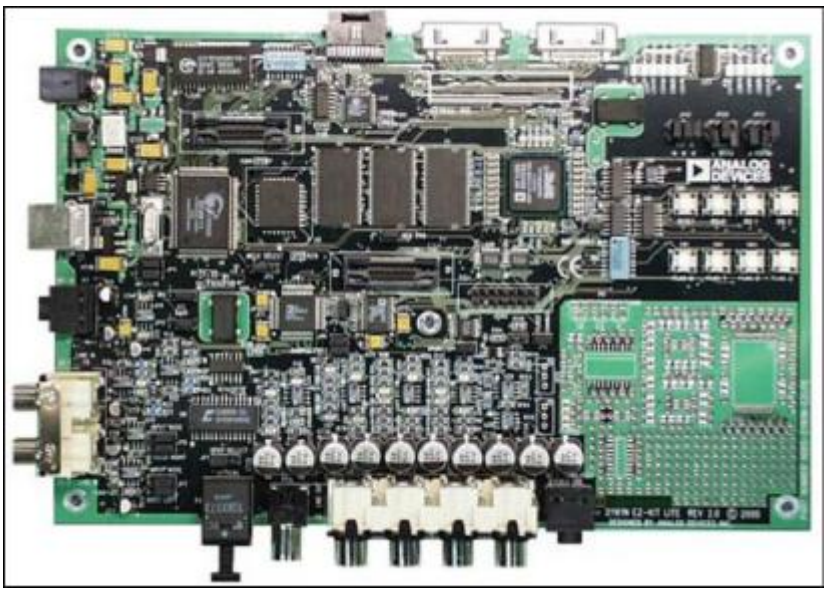

Figura 18. Plataforma ADSP-21161N EZ-KIT Lite

La plataforma ADSP-21161N EZ-KIT Lite, contiene los siguientes componentes:

1. ADSP-21161 SIMD SHARC DSP, funcionando a 100MHz
2. Memoria:
    - 1M-bit memoria en el chip
    - 1Mx48 SDRAM, funcionando a 100MHz
    - 512Kx8bit Memoria Flash
    - Cypress CY7C6403 EZ-USB Microcontroller (16 bit Host) conectado al Puerto JTAG, proporcionando un emulador JTAG, el cual funciona a través del puerto USB del PC.
    - AD1836 códec (24 bit, 96kHz, 4 ADCs, 6DACs)
    - Crystal CS8414 24-bit, 96kHz SP/DIF receiver
    - Un micrófono stereo I/P Jack, una entrada de línea RCA I/P Jack, 8 RCA de salida, entrada óptica o RCA para señales de audio de entrada.
    - Conector de emulación EZ-ICE JTAG.
    - Conectores de expansión (EP, SPORTS, Link Ports)
    - 6 leds de salida, 4 pulsadores de entrada (flags), 3 pulsadores de interrupciones.

En dicha plataforma, se hace uso de un códec, AD1836, el cual da flexibilidad al sistema para la captura y reproducción de la calidad profesional 24 bits de audio,

proporcionando la capacidad de procesamiento de múltiples señales de audio digital de alta fidelidad simultáneamente.

El códec AD1836 de 24 bits, el cual funciona en un sistema estéreo a 5V, proporciona 3 DAC´s estéreos y 2 DAC´s estéreos. Los ADC´s y DAC´s, son capaces de mantener 105dB de SNR[2] y de rango dinámico.

La capacidad de mantener la integridad de la calidad de la señal a 24 bit/105dB se puede lograr a través del procesamiento de audio de 32 bits sin recurrir a la aritmética de doble precisión, y aumento de rendimiento de 2x para muchos estándares de algoritmos DSP, tales como FIR, IIR y FFT.

El DSP, tiene la capacidad de procesamiento de señales de audio de 24 bits con el uso de SIMD[3].

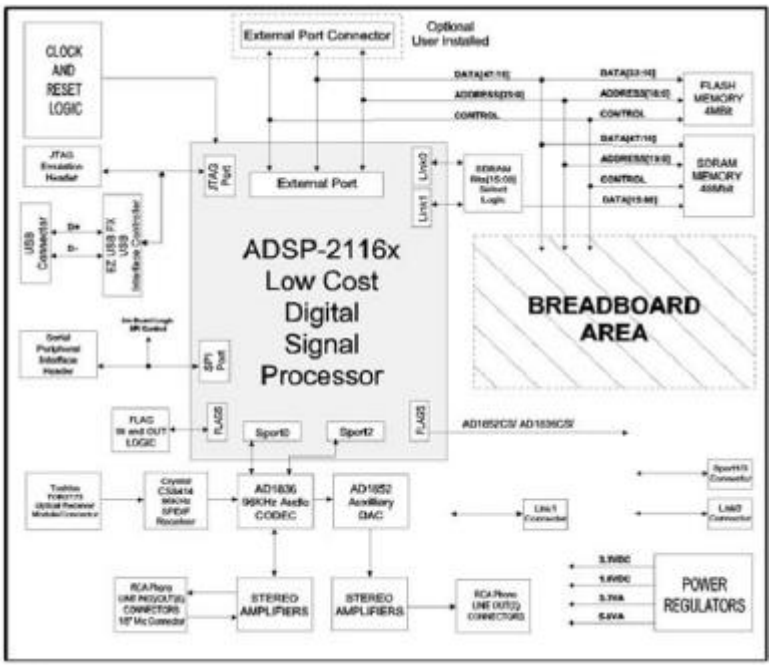

Figura 19. Diagrama de bloques de ADDS-21161N EZ-KIT LITE

La pareja AD1836/ADSP-21161, cumple los requisitos de alta fidelidad de audio para las nuevas aplicaciones emergentes de audio, y ofrece muchas ventajas para bajo costo de alta fidelidad de plataforma de audio, incluyendo:

[2] Relación señal - ruido
[3] **S**ingle **I**nstruction, **M**ultiple **D**ata, técnica empleada para conseguir paralelismo a nivel de datos.

a. **Compatibilidad del modo serie TDM del AD1836 con el modo serie multicanal (TDM)[4] del ADSP-21161:** este modo permite a un protocolo de sincronización que cada trama TDM genera ocho slots de tiempo, que son a su vez 32 bits por slot. La aproximación TDM permite el uso del puerto serie y de la cadena de DMA junto con el ADSP-21161 en el modo TDM de operación.
b. **Frecuencia de muestreo soportada para 24 bits, 48kHz/96kHz:** el AD1836 puede generar una frecuencia de muestreo de 48kHz o 96kHz, permitiendo la capacidad de grabar y reproducir audio de alta calidad.
c. **Alta calidad de 105dB de rango dinámico y SNR en los ADC's y DAC's:** superar la calidad del sonido que ofrece los 16 bits de conversión (96dB máximo). La capacidad de mantener los 105dB de la calidad de la señal, normalmente se suele encontrar en sistemas de audio profesional.
d. **Punto fijo de 32 bits SIMD, o 32/40 bits de punto flotante de procesamiento de audio de señales digitales de 24 bits a 600Mflops[5] de pico, 400Mflops de rendimiento sostenido:** la arquitectura del ADSP-21161 en el modo de SIMD, permite el procesamiento de señales de datos multicanal simultáneamente por medio de dos idénticas unidades computacionales, proporcionando la capacidad de mantener la integridad de la señal de 24-bit, garantizando de esta manera que cualquier error de cuantificación producidas durante las operaciones aritméticas, sean más bajos que los 105dB de ruido del AD1836, tanto ADC's como DAC's.

3.1.9. LOS BENEFICIOS DE USAR 32 BITS PARA EL PROCESAMIENTO DE AUDIO DE 24 BITS "DE CALIDAD PROFESIONAL" DE AUDIO

Hoy en día, lso 16 bits a 44,1kHz de audio digital sigue siendo el estándar para audio de alta calidad en la mayoría de las aplicaciones actuales, tales como CD, DAT y audio del PC.

Debido a recientes tecnologías y mejoras en el conocimiento de la audición humana, se ha creado una mayor demanda en la longitud de palabra en el sector profesional.

La sensibilidad del oído humano, es tal que el rango dinámico entre la zona más tranquila de sonido que la persona puede detectar y la máxima que puede experimentar es de

[4] **T**ime **D**ivision **M**ultiplexing, es un tipo de multiplexación digital, en el cual dos o más cadenas de bits, o señales son aparente y simultáneamente transferidas como sub-canal en un canal de comunicaciones, pero físicamente estas toman turnos sobre el canal.
[5] Flops: mediciones de rendimiento, en sus siglas en ingles **f**loating **p**oint **o**perations **p**er **s**econd, que traducido significa, operaciones de coma flotante por segundo.

120dB. La calidad de audio de 16bits, ya no se corresponde con la máxima calidad de audio de un sistema.

La tecnología digital ha avanzado tanto, que ya se pueden realizar grabaciones de hasta 120dB, pero un CD no puede llegar a esta precisión debido a los 16bits de longitud. Por lo que se ha desarrollado nuevos productos usando 24 bits de conversión y 96kHz de frecuencia de muestreo (DVD).

De hecho, hay tres tendencias que se pueden identificar con la influencia de la generación de formatos de audio digital, que se establecen para reemplazar el CD de audio digital:

- Alta resolución por palabra 20 bits o 24 bits
- Alta frecuencia de muestreo 96kHz
- Más canales de audio

El códec AD1836 de Analog Devices, ofrece 24 bits, una frecuencia de 96kHz, audio multicanal y reúne muchos requisitos en el mercado de audio profesional, de consumo y automoción por ejemplo. El AD1836 tiene una resolución de 24 bits y excede los 80 y 96dB de rango dinámico con los 16bits disponibles.

La razón para utilizar estos convertidores de mayor precisión es para tener un procesamiento de audio más limpio:

- Linealidad
- Incrementar el SNR y rango dinámico

Consideremos ahora el tamaño de la palabra DSP con respecto al tamaño de la palabra del convertidor para el procesamiento de muestras de audio digital.

La siguiente figura muestra los rangos dinámicos del DSP, con un ancho de palabra de 16,24 y 32 bits (asumiendo datos fraccionales de punto fijo). Se asume 6dB por bit, para 16bits tiene un rango dinámico de 96dB, para 24 bits soporta 144dB y para 32bits soporta hasta 196dB.

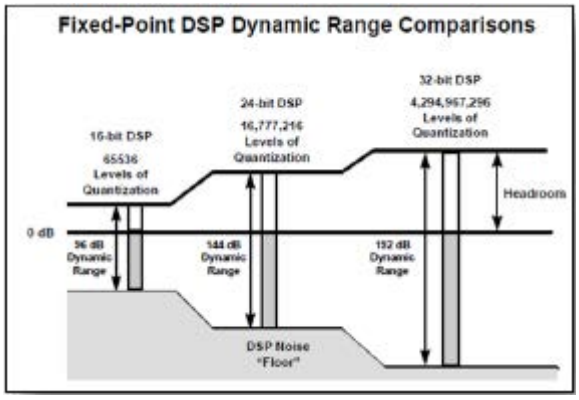

Figura 20. Rangos dinámicos del DSP con diferentes anchos de palabra.

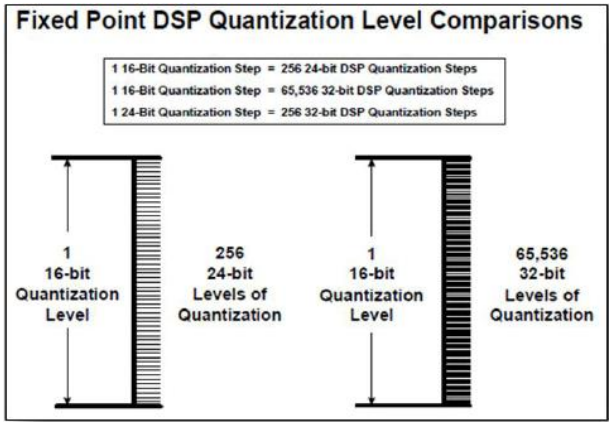

Figura 21. Niveles de cuantificación.

El algoritmo de procesamiento digital de funcionamiento sobre las muestras de entrada del A/D, deberían ser diseñados para prevenir el procesado de ruido. Para un filtro digital, la resolución del sistema debe ser considerablemente mayor que la señal de entrada, ya que los errores introducidos en los cálculos aritméticos, son más pequeños que la precisión del ADC y DAC. Para que el DSP mantenga el SNR establecido por los convertidores A/D, los cálculos del DSP requieren un procesamiento de mayor precisión.

La siguiente figura muestra cómo un mayor tamaño de la palabra de DSP, permite señales de las más alta calidad en la salida del DAC AD1836, cuando las señales de audio se procesan a 105dB de rango dinámico y SNR.

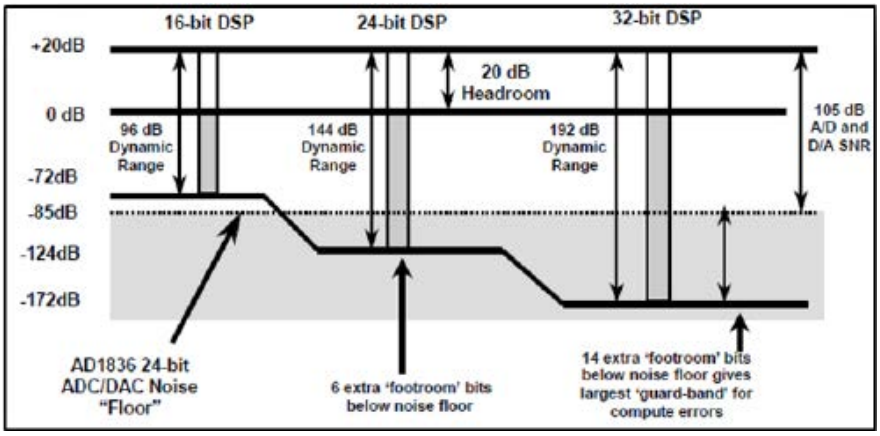

Figura 22. Calidad de la señal de salida en función del ancho de palabra.

$$\mathbf{SNR_{A/D(RMS)}(dB) = 6.02n + 1.76\ dB}$$

$$Dynamic\ Range(dB) = 6.02n + 1.76\ dB \cong 6n$$

El DSP de 16 bits, no tiene la precisión para representar palabras con como el proceso de las muestras de 24bits (a menos que se utilicen rutinas de doble precisión).

Para un DSP de 24 bits, hay poco margen para el error en los cálculos aritméticos. No puede ser adecuada para el procesamiento de precisión de las muestras de 24 bits debido a errores de truncamiento y redondeo (sin el uso de doble precisión).

El ADSP-21161 de 32 bits, tiene 14 bits por debajo del nivel de ruido, lo que permite la flexibilidad de un mayor SNR.

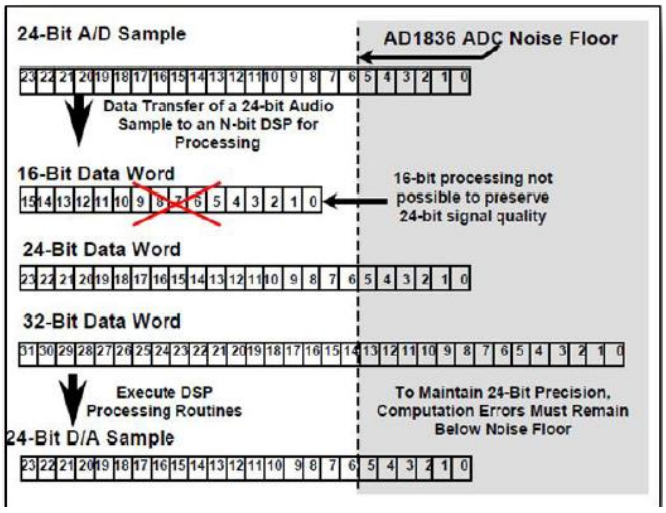

Figura 23. Representación de los bits que son adecuados para el procesamiento de 24 bits de muestra.

Un DSP de 24 bits, no es capaz de procesar adecuadamente las muestras de 24bits, sin recurrir a la doble precisión, especialmente para los algoritmos recursivos IIR de segundo orden.

La arquitectura SIMD del ADSP-21161, permite la codificación eficiente de algoritmos de audio, tomando ventaja de dos juegos idénticos de unidades de cómputo.

Cada elemento del proceso, da al DSP la capacidad de procesar de manera eficiente algoritmos complejos simultáneamente, mediante una operación en los datos del canal izquierdo, en un elemento de proceso de datos; y el canal derecho en el elemento de otro proceso.

ADSP-21161 incluye mejoras para realizar cálculos de audio en tiempo real:

- Soporta 32 bits en punto fijo y 32/40 bits en punto flotante.
- Rápida y flexible aritmética con el uso de 2 núcleos de computación. Permitiendo algoritmos como FFT, filtros FIR e IIR.
- Rango dinámico extendido para el cálculo ampliado de suma de productos con cuatro acumuladores de punto fijo de 80 bits.
- Buffer circular que soporta hasta 32 líneas de retraso.
- Eficientes bucles a través de un secuenciador de programa optimizado.
- 1 Mb SRAM en el chip, y un controlador de SDRAM en el chip para almacenamiento.
- 4 SPORTS con soporte de I2S, que pueden ser programados como transmisores o receptores.

Para preservar la calidad de las muestras de 24 bits, se requiere un minimo de 32 bits durante el procesamiento de la señal. Con 24 bits de datos de audio del AD1836, los 32 bits de procesamiento del ADSP-21161, es especialmente útil para el procesamiento recurrente con filtros IIR. Con un DSP de 32 bits, como el ADSP-21161, proporcionando 14 bits por debajo del ruido de fondo, los errores de cuantificación se tienen que acumular a 86dB.

Para mantener una alta calidad de audio de la señal, muy por encima de los 105dB de ruido del AD1836, el ADSP-21161 EZ-KIT Lite, es un sistema de diseño de referencia excelente que permite al programador del DSP, implementar algoritmos de tal manera que, todos los

cálculos del DSP tienen una precisión más alta que la longitud en bits de los datos de entrada cuantificados, utilizando cualquiera de los puntos fijos de 32 bits o 32-bit/40bits de punto flotante.

3.1.10. AD1836 Y ADSP 21161-N: INTERFAZ DE COMUNICACIÓN

Puesto que el DSP es un procesador digital, la señal física introducida por el canal de entrada ha de ser convertida para poder trabajar con ella. Por ello se utiliza el AD1836, el cual se encarga de la conversión a digital de la señal de entrada y de su posterior reconstrucción una vez "tratada" por el DSP, para luego ser entregada a los canales de salida.

El AD1836 posee 6 DAC's y 4 ADC's, una resolución de 24 bits y puede trabajar con frecuencias de muestreo de hasta 96kHz. Además, soporta conexiones I2S (para audio digital) con el DSP.

La comunicación con el ADSP 21161-N se hace mediante el protocolo de transmisión es que el AD1836, trabajando en este protocolo, puede generar como máximo una frecuencia de muestreo de 48kHz, ya que para generar 96kHz, se debería trabajar con una conexión I2S (Audio digital). Por lo tanto, queda establecido que, en este proyecto, la comunicación entre el ADSP 21161-N y el AD1836, será definido por una frecuencia de muestreo de 48kHz.

Para generar su reloj interno, el AD1836 hace uso de un cristal de cuarzo externo de 12.288MHz. Con el cual, la mayor frecuencia de muestreo que puede generar es de 48kHz. Así, esta es otra de las razones por las que la frecuencia de muestreo con la que hemos trabajado, es de 48kHz, ya que si hubiésemos querido usar o implementar una mayor frecuencia de muestreo (96kHz), el reloj externo a usar, debió ser de 24.576MHz.

Puesto que en el presente proyecto no se trabajará con audio digital, no tendremos en cuenta las conexiones I2S.

La comunicación entre el ADSP 21161-N y el AD1836 puede establecerse mediante una interfaz SPI (Serial Peripheral Interface) o bien a través de la interfaz serie SPORT.

La interfaz de comunicación entre ambos dispositivos se muestra en la siguiente figura:

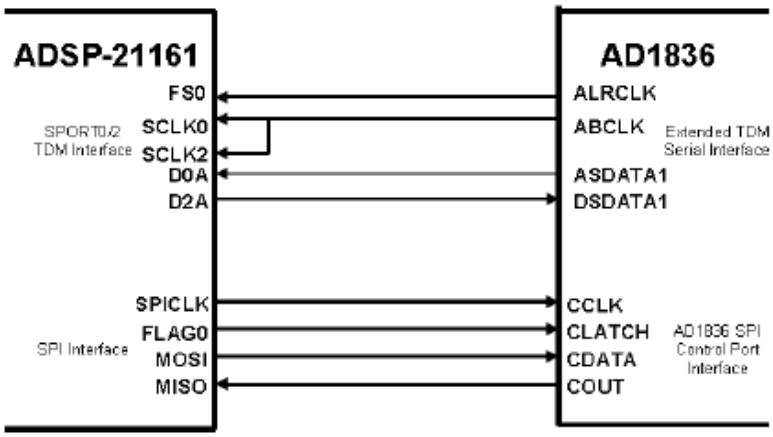

Figura 24. Interconexión entre ADSP-21161-N y AD1836

Como se puede ver en la figura, los datos son transmitidos a través de la conexión entre la interfaz TDM del AD1836 y la interfaz de conexión serie SPORT entre ambos dispositivos. Mientras, toda la información de control/estado se transmite mediante la interfaz SPI.

Es importante conocer, al menos, cómo son transmitidos los datos de un dispositivo a otro. Para ello, a continuación describiremos la función de cada uno de los pines de la interfaz de transmisión de datos.

A través del pin **ABCLK**, el AD1836 le transmite al ADSP 21161-N el reloj de 48kHz para poder procesar las tramas de 8 “slots” (fragmentos), de 32 bits cada uno, en las que se transmite la información (entrante y saliente).

El pin **ALRCLK** (llamado también FSTDM), es para la sincronización de las tramas.

Los pines **DSDATA1** (**D2A**) y **ASDATA1** (**D0A**) son la entrada y salida de datos del AD1836.

A modo aclaratorio, en las siguientes figuras se muestran las tramas TDM de entrada y de salida en una comunicación entre el AD1836 y el ADSP 21161-N:

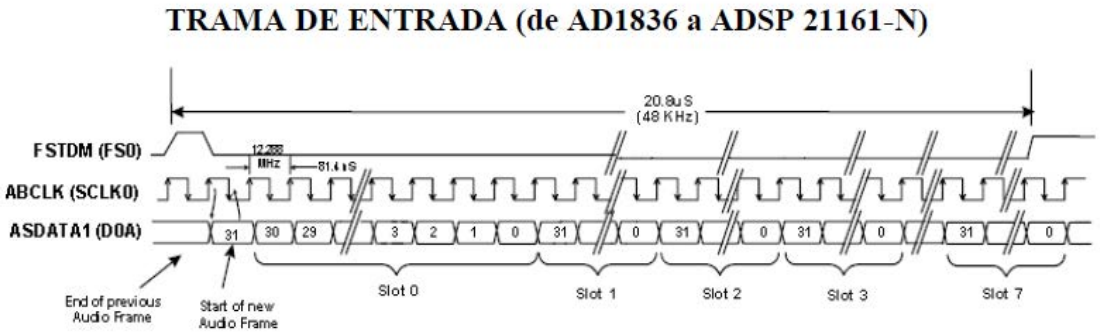

Figura 25. Trama TDM de entrada de datos.

Como se puede observar en la figura 25, la sincronización entre ambos dispositivos se produce con una transición de nivel bajo a nivel alto del pin **FSTDM**. Una vez sincronizados, el primer flanco de bajada del reloj generado por **ABCLK** es la confirmación para comenzar a transmitir datos. En el siguiente flanco de subida de **ABLCK** se empieza a transmitir el primer bit del "Slot 0" (primera muestra) y en el siguiente flanco de bajada el bit llega al DSP y es muestreado. De esta manera se hace la transmisión y muestreo del resto de bits de esta muestra y las 7 siguientes.

Además, los bits de todas las muestras que salen de **ASDATA1** están "justificados a la izquierda", lo cual quiere decir que los 24 bits que conforman cada muestra son los primeros 23 bits de cada "slot" de 32, siendo los últimos 8 bits "rellenados" con ceros para indicar que son datos no válidos.

En la siguiente figura se muestra el inicio de la trama de entrada más detallado:

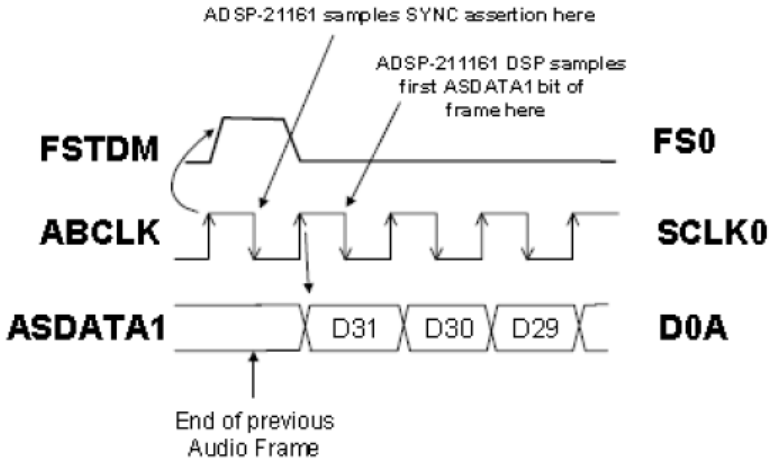

Figura 26. Inicio detallado de la trama TDM de entrada

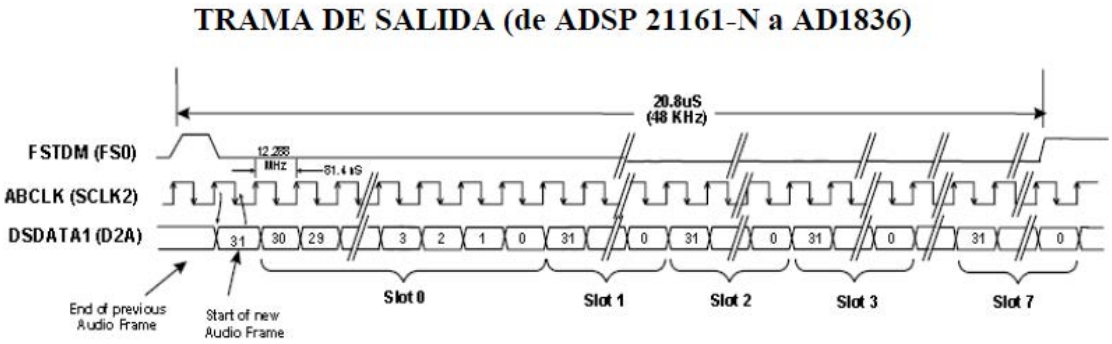

Figura 27. Trama TDM de salida de datos

Según muestra la figura 27, la transmisión de la trama desde el DSP hacia el AD1836 es muy parecida a la transmisión de la trama anterior. La sincronización entre ambos dispositivos se produce con una transición de nivel bajo a nivel alto del pin **FSTDM**. Una vez sincronizados, el primer flanco de bajada del reloj generado por **ABCLK** es la confirmación para comenzar a transmitir datos. En el siguiente flanco de subida de **ABCLK** se empieza a muestrear el primer bit del "Slot 0" (primera muestra) y en el siguiente flanco de bajada el bit es transmitido por el DSP. De esta manera se hace la transmisión y muestreo del resto de bits de esta muestra y las 7 siguientes.

Además, los bits de todas las muestras que salen de **DSDATA1** están "justificados a la izquierda", lo cual quiere decir que los 24 bits que conforman cada muestra son los primeros 24 bits de cada "slot" de 32, siendo los últimos 8 bits "rellenados" con ceros para indicar que son datos no válidos.

En la siguiente figura se muestra el inicio de la trama de entrada más detallado:

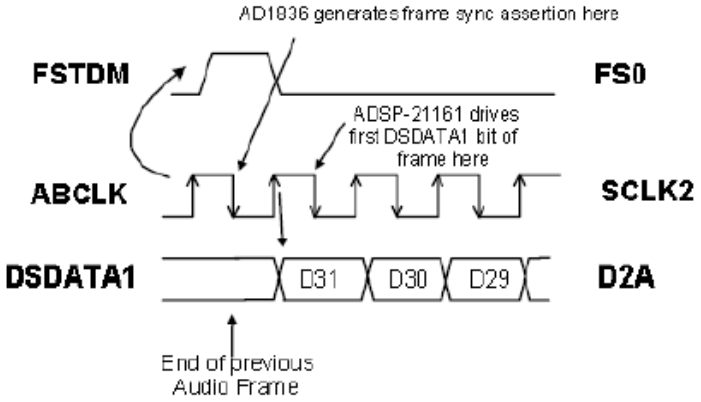

Figura 28. Inicio detallado de la trama TDM de salida

Finalmente, con el fin de que se aprecie una visión más "física" y clara de todo el "KIT de evaluación" y de la interconexión entre conectores, ADSP 21161-N y AD1836, podemos observar la siguiente figura:

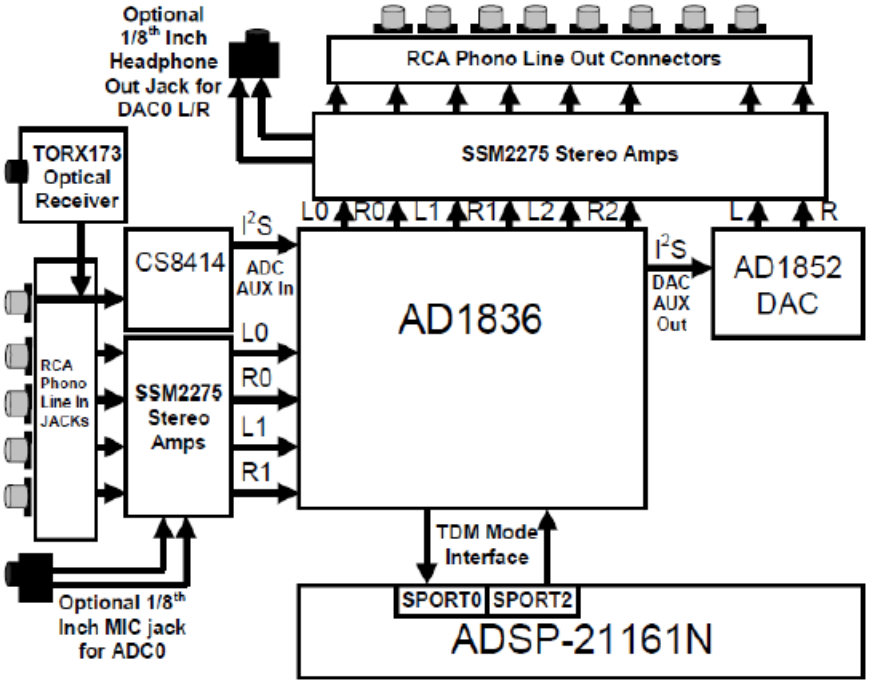

Figura 29. Apariencia de conexiones entre recursos de la placa

4. HERRAMIENTA SOFTWARE

4.1. VISUAL DSP++4.5

El software que utilizamos para la ejecución de nuestro proyecto, será el Visual DSP ++4.5, el cual es un entorno que se encarga de cargar y ejecutar los programas en el dispositivo DSP de la placa. Basándose los programas a implementar, en el lenguaje de programación C++.

Dicha herramienta nos permite realizar una serie de acciones, de las cuales solo expondremos las que usamos al realizar el presente proyecto.

Dichas acciones son las siguientes:

- Crear sesión: Se refiere a la placa que se "conectará" con el software instalado. En nuestro caso, nos referimos al "KIT de evaluación" ADSP 21161-N. Cabe destacar en este punto que la conexión "física" entre el PC y el KIT se describirá más adelante. Resaltar que usaremos un DSP 21161 de la familia SHARC, siendo el tipo de conexión EZ KIT Lite y la plataforma ADSP-21161 EZ-KIT Lite

- Crear proyecto: Se podría decir que esta parte es la más importante y fundamental en la realización de este proyecto, ya que aquí es donde se crean los programas que se cargarán en el "KIT de evaluación".

- Adjuntar archivos .c o .h: Dicha acción se podrá realizar bajo una condición, y es que antes de adjuntar los ficheros y/o archivos con extensión .c o .h, se necesita estar sobre la carpeta de trabajo.

- Comandos más importantes: Existen una serie de "botones" de acceso rápido a los comandos más usados en el entorno Visual DSP++, como por ejemplo:

    - Construir y cargar proyectos (build)
    - Ejecutar (run)
    - Parar (halt)
    - Realizar un reset a la placa (reset)

- Crear gráficas: En un proyecto, al ejecutar un programa podemos tener la necesidad de observar el efecto del código creado sobre algunas variables. Para poder visualizar estos efectos, podemos representar gráficamente estas variables, con la ayuda del "Plot", la cual nos sirve para observar el comportamiento de las señales.

4.2. MATLAB

Además del software que detallamos con anterioridad, el Visual DSP++ 4.5, utilizamos el programa Matlab.

Este software nos permite simular y ejercer a modo de práctica, el comportamiento que tendrán nuestros filtros ante diferentes tipos de excitación y ante distintos ordenes de los filtros que usemos.

A continuación pasaremos a exponer ciertas utilidades de este software:

- Crear archivos .m: Este tipo de ficheros, permite almacenar las sentencias necesarias para luego ser ejecutadas.
- Funciones más utilizadas: Debido a que para la realización de filtros digitales, se necesito una serie de funciones, pasaremos a detallar que parámetros se pasan a cada función y para que se usó.
    - Función *butter*: Utilizada para crear los coeficientes de un filtro de Butterworth. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).
      Ejemplo: [B,A]=butter(N,Wn,'low')
      Donde:
      B= coeficientes del numerador
      A= coeficientes del denominador
      N= orden del filtro
      Wn= frecuencia normalizada, que sale de la conversión de la frecuencia analógica a frecuencia digital:
      Wn=(2*pi*fc/Fm); f= frecuencia de corte analógica
      Fm= frecuencia de muestreo
    - Función *cheby1*: Utilizada para crear los coeficientes de un filtro de Chebyshev del tipo 1. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte, la tasa pico a pico de la banda de paso y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).
      Ejemplo: [B,A]=cheby1(N,R,Wn,'low')
      Donde:
      B= coeficientes del numerador
      A= coeficientes del denominador
      N= orden del filtro
      R= tasa pico a pico de la banda de paso

Wn= frecuencia normalizada, que sale de la conversión de la frecuencia analógica a frecuencia digital:

Wn=(2*pi*fc/Fm); f= frecuencia de corte analógica

Fm= frecuencia de muestreo

- Función *cheby2*: Utilizada para crear los coeficientes de un filtro de Chebyshev del tipo 2. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte, la tasa pico a pico de la banda de paso y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).
  Ejemplo: [B,A]=cheby2(N,R,Wn,'low')
  Donde:
  B= coeficientes del numerador
  A= coeficientes del denominador
  N= orden del filtro
  R= tasa pico a pico de la banda de paso
  Wn= frecuencia normalizada, que sale de la conversión de la frecuencia analógica a frecuencia digital:
  Wn=(2*pi*fc/Fm); f= frecuencia de corte analógica
  Fm= frecuencia de muestreo
- Función *ellip*: Utilizada para crear los coeficientes de un filtro Elíptico. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte, la tasa pico a pico de la banda de paso, la mínima atenuación de la banda de rechazo y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).
  Ejemplo: [B,A]=ellip(N,Rp,Rs,Wn,'low')
  Donde:
  B= coeficientes del numerador
  A= coeficientes del denominador
  N= orden del filtro
  Rp= tasa pico a pico de la banda de paso
  Rs= mínima atenuación de la banda de rechazo
  Wn= frecuencia normalizada, que sale de la conversión de la frecuencia analógica a frecuencia digital:
  Wn=(2*pi*fc/Fm); f= frecuencia de corte analógica
  Fm= frecuencia de muestreo

- Función *adaptfilt.lms*: Utilizada para crear los objetos de un filtro adaptativo LMS. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte, la tasa pico a pico de la banda de paso, la mínima atenuación de la banda de rechazo y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).

  Ejemplo: `ha = adaptfilt.lms(l,step,leakage,coeffs,states)`

  Donde:

  ha= objetos del filtro adaptativo LMS

  l= longitud del filtro adaptativo

  `step`= tamaño de paso

  leakage= factor de fuga

  Coeffs= vector de coeficientes iniciales del filtro

  States= vector de estados iniciales del filtro

- Función *fir1*: Utilizada para crear los coeficientes de un filtro FIR. Hay que indicarle el orden del filtro, la frecuencia normalizada de corte y el comportamiento deseado para el filtro (paso-bajo, paso-alto, paso-banda, banda-rechazada).

  Ejemplo: [B,A]=fir1(N,Wn,'low')

  Donde:

  B= coeficientes del numerador

  A= coeficientes del denominador, que al ser un filtro FIR será unitario,

  N= orden del filtro

  Wn= frecuencia normalizada, que sale de la conversión de la frecuencia analógica a frecuencia digital:

  Wn=(2*pi*fc/Fm); f= frecuencia de corte analógica

  Fm= frecuencia de muestreo

- Función *tf*: : Utilizada para crear la función de transferencia de un filtro a partir de sus coeficientes. Además, hay que indicarle el período de muestreo (1/fm).

  Ejemplo: H=tf(B,A,Tm)

  Donde:

  H= función de transferencia del filtro,

  B= coeficientes del numerador

  A= coeficientes del denominador,

Tm= período de muestreo

- Función *rlocus*: Utilizada para mostrar el lugar de las raíces de una determinada función de transferencia. Así podemos observar si nuestro sistema está dentro de los límites de la estabilidad.
  Ejemplo: rlocus(H)
  Donde:
  H= función de transferencia del filtro

4.3. COOL EDIT PRO

Esta herramienta es muy útil para crear audio digital. Con él pueden crear, mezclar y editar señales de audio de distintos tipos y frecuencias. Además, ofrece la posibilidad de generar ruido.

Para este proyecto es básico explicar cómo se pueden generar distintos componentes de audio y ruido en una sesión con Cool Edit Pro para poder realizar posteriormente el filtrado de las mismas. Por ello, a continuación explicaremos de forma detallada cómo crear dicha sesión en el programa y cómo generar e ir añadiéndole distintas componentes de audio, ya sean tonos o ruido.

4.3.1. CREAR UNA SESIÓN

Para empezar a añadir pistas con diferentes componentes de audio, lo primero que hay que hacer es iniciar una sesión. Para ello, en la interfaz de usuario, en la esquina superior izquierda observamos un "switch" para poder alternar entre ver y editar una sóla pista o ver y editar una sesión multitrack. Según la opción escogida, el "switch" cambia de apariencia:


Sesión Multitrack


Edición de una pista

De esta manera, si queremos crear una nueva sesión hemos de tener seleccionada la opción de **Sesión Multitrack**, ya que para cada posición del "switch" el menú *File* ofrece distintas opciones. Una vez marcada esta opción, nos vamos al menú *File* y seleccionamos la opción *New Session*.

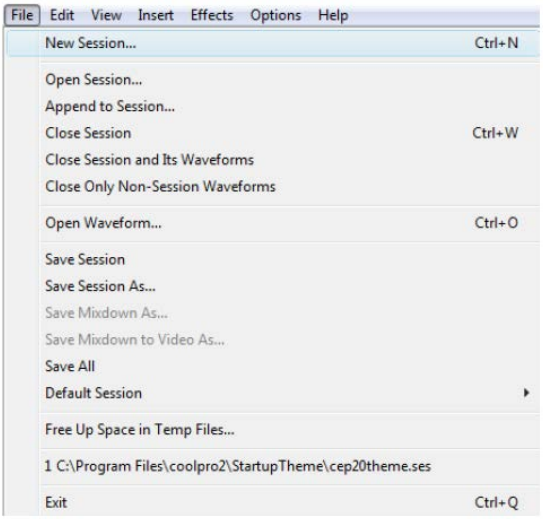

Figura 30. Crear una nueva sesión

Al seleccionar la opción de crear una sesión nueva, lo siguiente que nos pide el programa es que establezcamos la frecuencia de muestreo de la sesión.

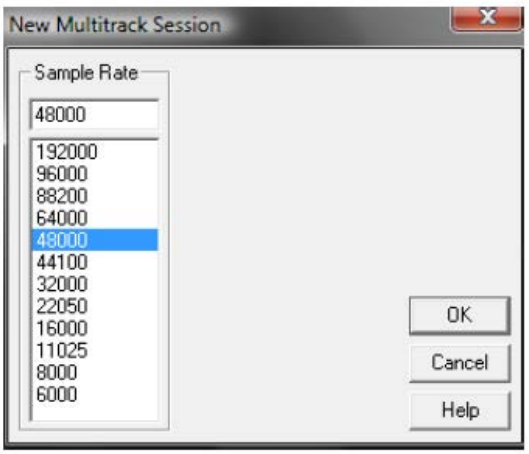

Figura 31. Seleccionar frecuencia de muestreo

Hasta aquí, ya hemos creado una nueva sesión multitrack en la que poder crear y añadir diversas componentes de audio (figura zz).

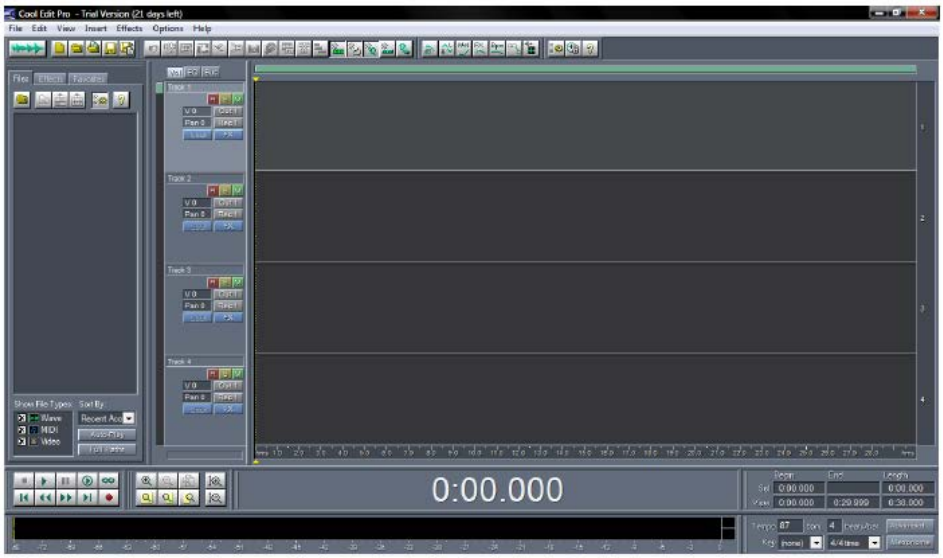

Figura 32. Sesión multitrack creada.

4.3.2. CREAR UNA PISTA DE AUDIO

Una vez creada la sesión, podemos empezar a generar distintas componentes e ir añadiéndolas. Para crear una pista, hemos de tener el “switch” en la opción **Edición de una pista**. Así, la apariencia de la interfaz de usuario es la siguiente:

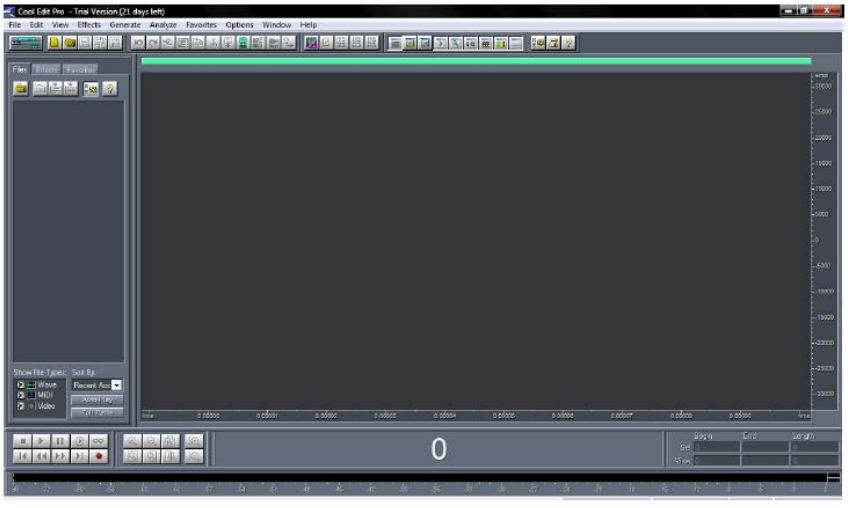

Figura 33. Interfaz de creación de una pista de audio

Como podemos observar, en la parte izquierda de la pantalla tenemos una “mini ventana” en la que aparecen tres solapas: *File, Effects y Favorites*. Si seleccionamos la pestaña *Effects* podemos ver cómo nos aparecen unos menús en la ventana debajo suya. De estos menús, el que nos interesa para crear componentes sencillas es el menú *Generate*.

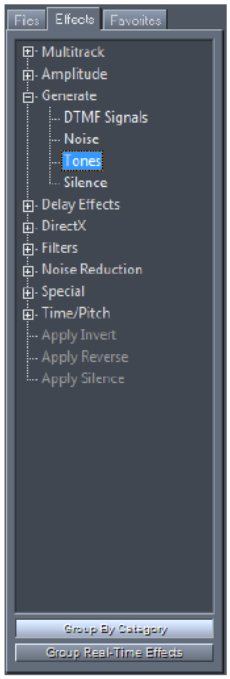

Figura 34. Generar un tono.

Una vez desplegado este menú, dentro de las opciones que nos aparecen debajo, podemos elegir el tipo de señal que queremos crear. Si queremos generar un tono, por ejemplo, seleccionaremos *Tones*. Escogido el tipo de forma de onda que queremos, el programa nos irá pidiendo una serie de parámetros: Frecuencia de Muestreo, señal Mono o Stereo, Resolución.

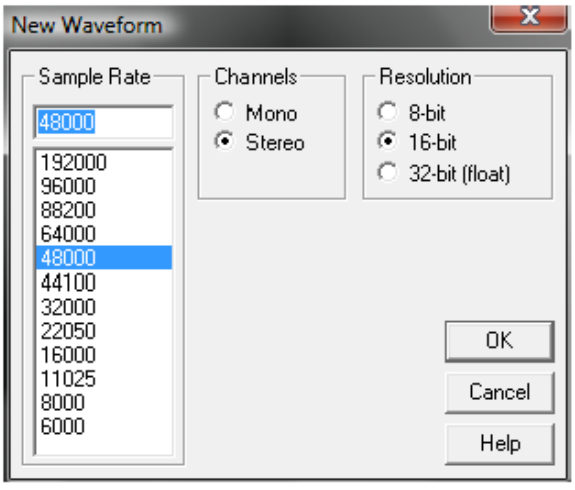

Figura 35. Frecuencia de muestreo, número de canales y resolución

La siguiente ventana que aparece es en la cual vamos a decidir parámetros como: según las dos solapas de la parte superior definiremos ajustes iniciales o finales; la base de

frecuencia de la señal; si queremos modular; si queremos que la señal sea un seno, una señal cuadrada, una señal triangular; las componentes que tendrá la señal generada y las frecuencias de dichas componentes; la duración del tono; si tendrá offset…

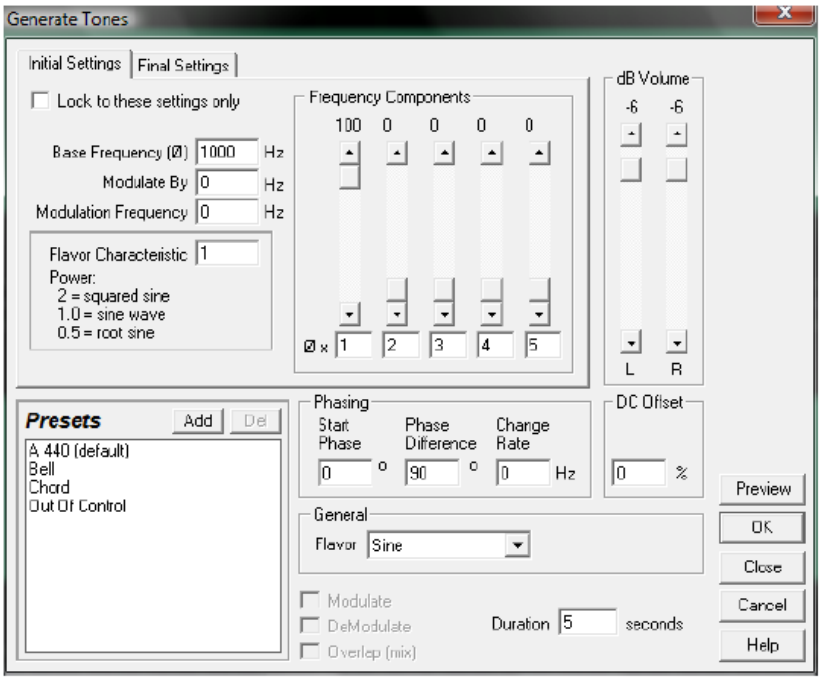

Figura 36. Parámetros de la pista

Si, después de haber definido todos los parámetros de la señal a nuestro gusto, seleccionaremos *OK* ya tendremos nuestra pista creada y su apariencia se nos muestra en la ventana de usuario (haciendo uso del zoom la veremos correctamente):

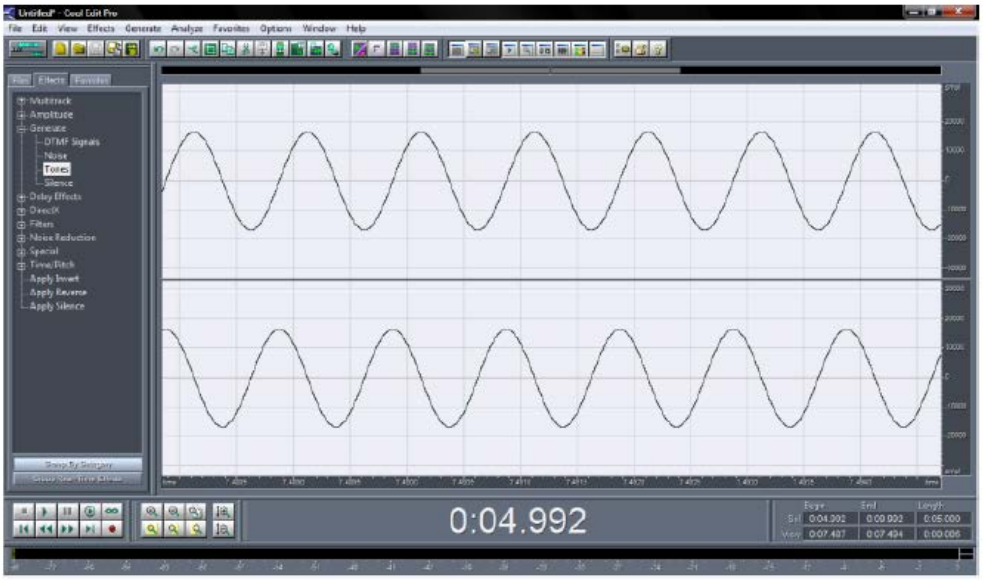

Figura 37. Pista creada

Así, podemos ir creando tantas pistas de audio como necesitamos para después poder unirlas a la sesión ya creada. Si queremos ir guardándolas, basta con ir seleccionando la opción *Save as* del menú *File*.

4.3.3. CREAR UNA PISTA DE RUIDO

Si queremos crear una componente en la que sólo aparezca ruido, simplemente hemos de volver a seleccionar la pestaña *Effects* (con el "switch" en **Edición de una pista**) y dentro de los menús que aparecen escoger *Noise*.

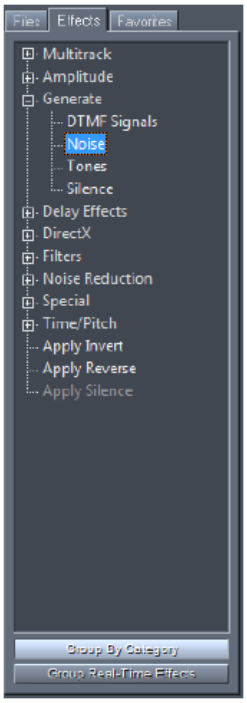

Figura 38. Generar pista de ruido

Después el programa nos volverá a pedir los parámetros a establecer, que en este caso serán más simples, como el tipo de ruido (Marrón, Rosa o Blanco) o la duración e intensidad del mismo.

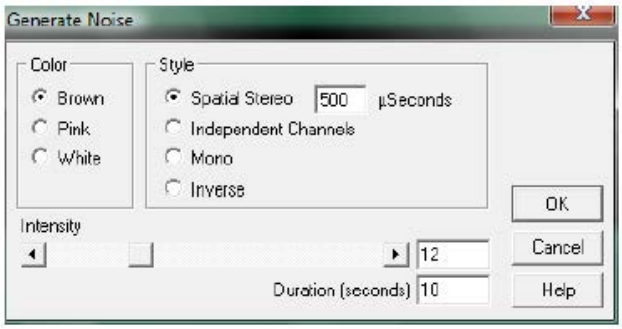

Figura 39. Tipo de ruido, duración e intensidad

Una vez escogido el tipo de ruido, para crearlo le damos a *OK* y podemos ver su apariencia en la interfaz de usuario (haciendo uso del zoom se puede ampliar ciertos sectores, apreciándose mejor).

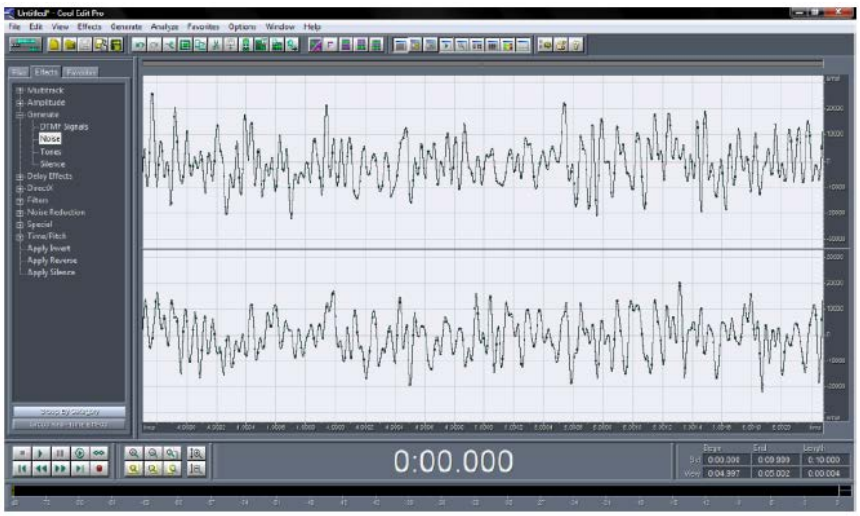

Figura 40. Pista de ruido generada

4.3.4. AÑADIR PISTAS DE AUDIO A UNA SESIÓN YA CREADA

Una vez generada las componentes deseadas, podemos ir añadiéndolas a la sesión principal para que queden mezcladas. Para ello, hemos de ir guardando cada pista creada seleccionando la opción *Save as* del menú *File*. Así, las vamos guardando como archivos de extensión .wav que podemos ir añadiendo a nuestra sesión. Por ejemplo, en la siguiente figura vemos cómo hemos guardado dos componentes: 1kHz y 10kHz.

Figura 41. Pistas creadas

Teniendo estos dos tonos, si colocamos el "switch" en **Sesión Multitrack** podemos arrastrar cada componente a la multisesión y ya tendremos una sesión con dos pistas añadidas. Para hacernos una idea del resultado de añadirlas, en la siguiente figura se muestra cómo queda la interfaz de usuario en **Sesión Multitrack:**

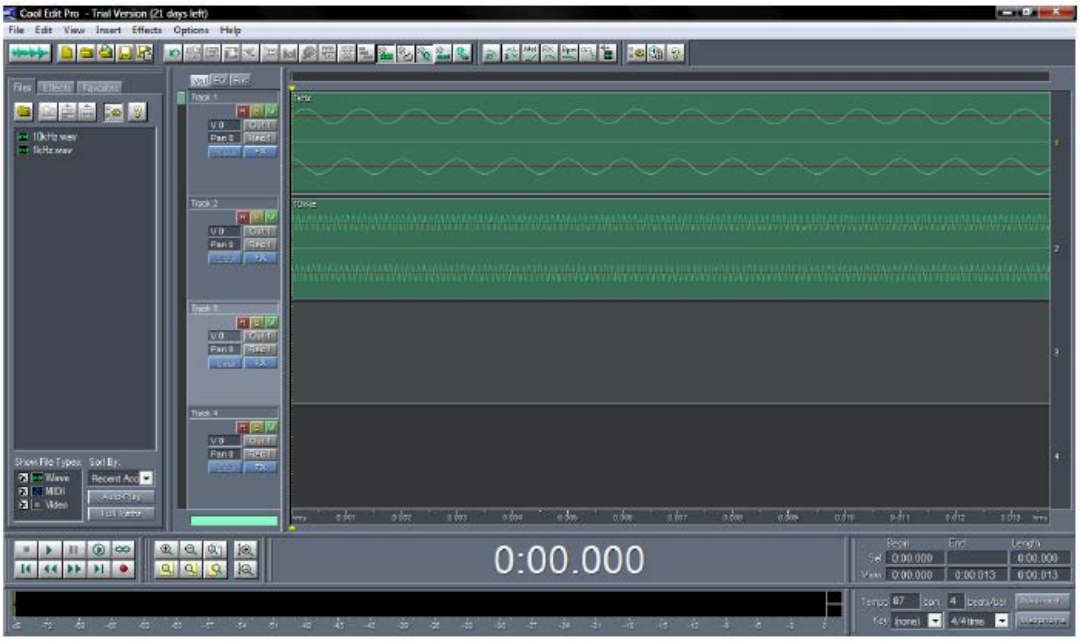

Figura 42. Sesión multitrack con varias pistas mezcladas

Finalmente, si queremos ejecutar la sesión lo único que tenemos que hacer es seleccionar la opción *Play*.

5. ACELERÓMETRO

Los acelerómetros son dispositivos para medir aceleración y vibración. Estos dispositivos convierten la aceleración de gravedad o movimiento, en una señal eléctrica analógica proporcional a la fuerza aplicada a un sistema, o u mecanismo sometido a vibración o aceleración. Esta señal analógica, indica en tiempo real, la aceleración instantánea del objeto sobre el cual el acelerómetro está montado.

$$F(t) = ma(t)$$

Estos dispositivos, los acelerómetros, miden la aceleración en unidades “g”. Siendo un “g” definido por la fuerza gravitacional de la tierra aplicada sobre el objeto o persona ($g=9.80665\ m/s^2$). Además resaltar que los acelerómetros son direccionales, es decir miden la aceleración en un solo eje.

Los acelerómetros pueden ser usados como un estándar de referencia, como objeto de test o como un objeto de calibración. Pueden llegar a tener una gran variedad de distintos tipos de sensores.

Tenemos por ejemplo:

- Sensores piezoeléctricos
- Servo sensores
- Sensores piezorresistivos
- MEMS

El que usamos en nuestras pruebas de laboratorio, para este proyecto son los sensores piezorresistivos, por lo que pasaremos a detallar un poco las características que estos presentan.

5.1. SENSORES DE ACELERACIÓN PIEZORRESISTIVOS

En estos tipos de acelerómetros sensoriales, también se utiliza la ecuación: $F(t) = ma(t)$. Sin embargo, no es un cristal que esté expuesto a la fuerza, pero sí un conjunto de uno, dos o cuatro medidores de tensión. Dichos medidores de tensión son expandidos y comprimidos alternativamente por la fuerza, resultando unas variaciones de resistencia, las cuales son transformadas en variaciones de resistencia. El puente está alimentado por una tensión continua estable.

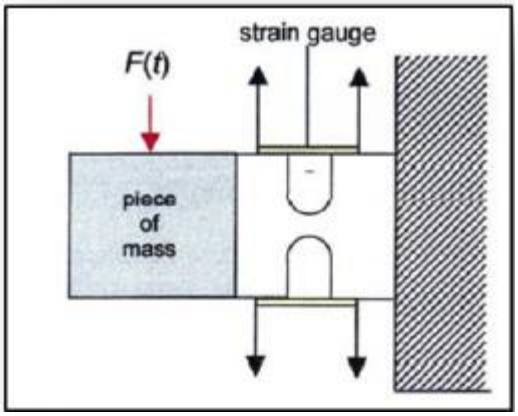

Figura 43. Acelerómetro piezorresistivo

El puente de resistencias (puente de Wheatstone), incluye dos cables adicionales (+Sens, -Sens), usados para medir la verdadera tensión de alimentación del puente. A, B, C y D son uno, dos o cuatro resistencias que dependen de la aceleración. +SIG y –SIG, representa la tensión de salida y, +EXC y –EXC, representan la excitación.

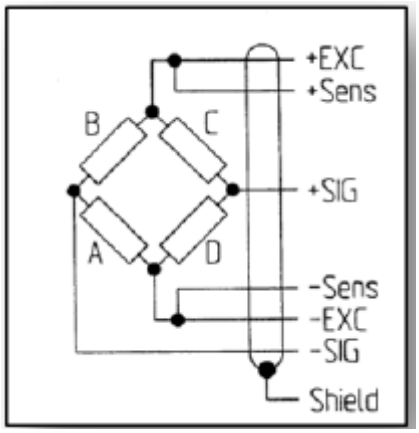

Figura 44. Puente de Wheatstone.

5.2. MÉTODOS DE CALIBRACIÓN

La calibración consiste en someter al acelerómetro patrón y al acelerómetro a una misma vibración de amplitud y frecuencias determinadas para comparar la lectura de aceleración. Tras haber obtenido los valores, se determina la sensibilidad del acelerómetro, su linealidad y su respuesta en frecuencia.

Existen dos métodos de calibración:

- Calibración primaria
- Calibración secundaria

5.2.1. CALIBRACIÓN PRIMARIA

El método de calibración primario o absoluto, es el utilizado por los laboratorios de calibración que disponen de patrones primarios para la calibración de acelerómetros de referencia.

El procedimiento más utilizado en los laboratorios primarios para la calibración de un acelerómetro de referencia, es el basado en interferometría[6] debido a su fiabilidad y gran exactitud. Sirve para medir de forma remota velocidades de superficie o vibraciones en puntos específicos de una estructura en vibración.

Consiste en someter al acelerómetro en calibración a una vibración sinusoidal mientras se registra el desplazamiento en el tiempo mediante un interferómetro láser situado

[6] La interferometría es una técnica que consiste en combinar la luz proveniente de diferentes receptores, telescopios o antenas de radio para obtener una imagen de mayor resolución.

perpendicular al eje de movimiento y así extraer la sensibilidad del acelerómetro y verificar su respuesta en amplitud y frecuencia.

Los acelerómetros se tienen que fijar en la superficie de prueba, los vibrómetros laser operan sin contacto y no son afectados por la superficie, ni por las condiciones ambientales, tales como la temperatura y/o presión.

El rango de medición de amplitud y frecuencia de un vibrómetro láser abarca desde señales en DC a señales de más de 300kHz. Puesto que, la salida es directamente proporcional a la velocidad instantánea de la superficie, esta técnica ofrece una alternativa a los acelerómetros de contacto para mediciones de movilidad.

Figura 45. Calibración primaria

1. Generador de vibraciones
2. Excitador de vibración
3. Amplificador de potencia
4. Interferómetro láser
5. Aislante de vibraciones
6. Acondicionamiento de la señal y adquisición de datos para el acelerómetro de referencia y el acelerómetro.

5.2.2. CALIBRACIÓN SECUNDARIA

La calibración secundaria de acelerómetros está basada en la calibración por comparación *back to back*.

Esta calibración, *back to back,* consiste en someter al acelerómetro y al acelerómetro de referencia a la misma señal de aceleración, y comparar sus salidas para obtener la sensibilidad del acelerómetro.

El procedimiento de calibración secundario o por comparación más habitual de acelerómetros de medio y alto rango de uso industrial, utiliza un sistema formado por un acelerómetro de referencia montado sobre un vibrador controlado por un generador de frecuencia sinusoidal y un amplificador de potencia. Sobre el acelerómetro de referencia se fija el acelerómetro en calibración para comparar directamente las señales producidas por ambos sensores ante determinadas vibraciones. El objetivo de este tipo de calibración también es el de obtener la sensibilidad del acelerómetro y conocer su respuesta en amplitud y frecuencia.

Mediante este tipo de calibración y dependiendo de los instrumentos utilizados, es posible alcanzar aceleraciones de hasta unos 50g de pico y frecuencias que pueden ir desde 10Hz hasta 50kHz. Éste es el método utilizado por nuestro laboratorio de calibración para realizar las calibraciones de los acelerómetros.

Una de las mayores ventajas que supone este tipo de calibración es la posibilidad de realizar un barrido de frecuencias para determinar si la sensibilidad se mantiene a lo largo de todo el ancho de banda del acelerómetro.

Las medidas realizadas de los distintos acelerómetros bajo prueba, fueron hechas con el sistema de calibración **CS18TF (SPEKTRA)**. Este sistema puede llevar a cabo calibraciones de sensores con o sin amplificadores, en el ranfo de frecuencias de 3Hz hasta 5kHz.

Figura 46. Calibración secundaria.

1. Generador de frecuencia sinusoidal
2. Amplificador de potencia
3. Excitador de la vibración

4. Acelerómetro de referencia
5. Acondicionamiento de la señal y adquisición de datos para el acelerómetro de referencia y el acelerómetro.

Este modo de calibración también cuenta con un módulo para acelerómetros piezorresistivos, el cual proporciona la señal de salida y también alimenta el puente de Wheatstone.

Figura 47. Modulo piezorresistivo.

5.3. DESCRIPCIÓN DE ACELERÓMETRO

El acelerómetro expuesto a continuación, es piezorresistivo, sobre el cual se ha realizado las medidas oportunas.

El acelerómetro medido fue el 4000A-002-060 de la serie A046144. Trabaja con una frecuencia de resonancia[7] de 700Hz y tiene una sensibilidad de 100 mV/g.

Todo cuerpo o sistema tiene una, o varias frecuencias características. Cuando un sistema es excitado a una de sus frecuencias características, su vibración es la máxima posible. El aumento de vibración se produce porque a estas frecuencias el sistema entra en resonancia.

A partir de ahora nos referiremos al acelerómetro como 4000A

5.4. ACELERÓMETRO 4000A

El acelerómetro 4000A, tiene una señal acondicionada que es ideal para bajas frecuencias en aplicaciones de monitoreo. Incorpora una segunda generación de sensores MEMS piezorresistivo. Se encuentra en una robusta carcasa de aluminio ideal para las pruebas de transporte y de instrumentación. La salida de la señal acondicionada incorpora una tensión

[7] La frecuencia de resonancia es aquella a la que la respuesta en frecuencia alcanza su máximo. Es decir, dada una entrada, se obtiene una salida máxima.

de referencia de 2.5V que ofrece al usuario una salida diferencial o la salida de un solo terminal. Presenta una respuesta en frecuencia de 200Hz.

Figura 48. Acelerómetro 4000ª

Características:

- Rango dinámico de ±2g hasta ±2000g
- Alta gama de protección
- Señal de salida acondicionada
- Bajo consumo de energía
- Peso ligero
- Amortiguación de gas
- Cable integral-opciones de conector

Aplicaciones:

- Baja frecuencia de monitoreo
- Transporte
- Medidas de vibración
- Pruebas e instrumentación
- Control de la maquina
- Análisis de movimiento
- Inclinación

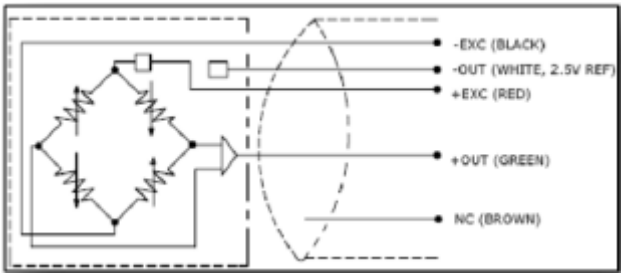

Figura 49. Conexión del acelerómetro 4000A

6. DISEÑO DE FILTROS DIGITALES CON MATLAB

Para realizar el diseño de filtros con Matlab, se emplearán las funciones que explicamos en el apartado dedicado al software de Matlab.

Además necesitaremos una serie de señales de entrada para excitar los distintos filtros a diseñar.

Con estos prerrequisitos podremos diseñar todos los filtros relatados en apartados anteriores, para luego pasar a implementarlos en la tarjeta de pruebas, excitarla con señales producidas a través del Cool Edit Pro y finalmente utilizaremos dichos filtros con señales rescatadas de un acelerómetro.

Dicho esto, pasaremos al diseño de los filtros Butterworth, Chebyshev del tipo 1 y 2 y elíptico, que se mostraran primeramente en matlab y luego lo introduciremos en el DSP. No haremos mucho énfasis en esta parte, ya que el objetivo principal es el diseño de los filtros adaptativos, el filtro adaptativo LMS (Least Mean Square), y su funcionamiento sobre la tarjeta de pruebas con señales tanto creadas por el Cool Edit Pro, como también aquellas provocadas por el acelerómetro.

Por lo tanto comenzaremos con el análisis y comportamientos de los filtros Butterworth, Chebyshev y Eliptico.

6.1. FILTRO BUTTERWORTH

Para el desarrollo de este filtro se utilizó el siguiente comando en Matlab, [B,A]=butter(N,Wn). En el Anexo A, se puede observar los comandos utilizados para la implementación y estudio del filtro buterworth.

Como se puede observar en las figura 50 y 51, podemos ver que el comportamiento de nuestro filtro es el indicado.

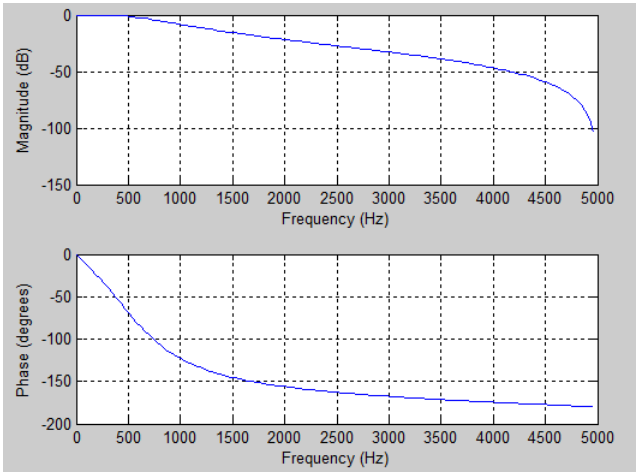

Figura 50. Grafica en decibeles y en fase de un filtro Butterworth paso bajo de segundo orden

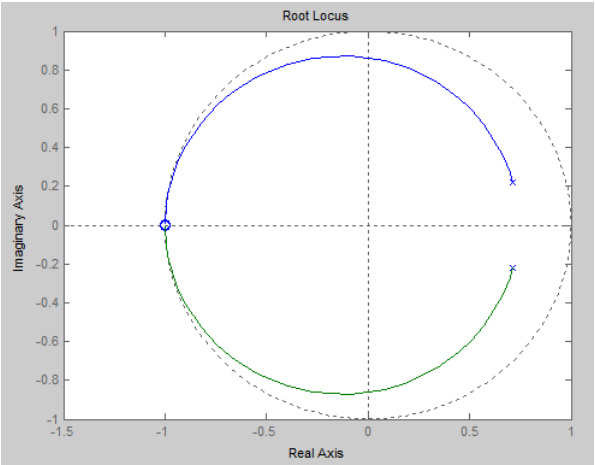

Figura 51. Diagrama de polos y ceros del filtro Butterworth paso bajo de segundo orden.

En las figuras previas (figura 50 y figura 51), tomamos como ejemplo un filtro paso bajo de segundo orden con frecuencia de corte en un kilo Herzio y frecuencia de muestreo de cuarenta y ocho kilo Herzios.

Como se observa en la figura 50, podemos observar que la atenuación cesa a partir de 1KHz aproximadamente; de la misma manera se puede observar que el diagrama de fases tiene una respuesta lineal.

A continuación se mostrará el comportamiento de un filtro paso alto, paso banda y paso banda eliminada:

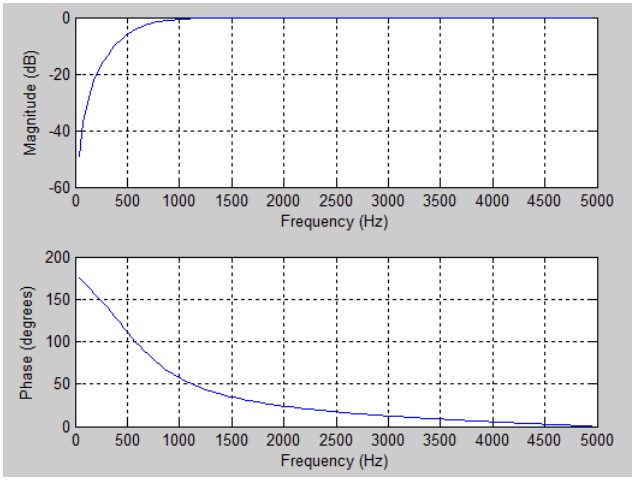

Figura 52. en decibeles y en fase de un filtro Butterworth paso alto de segundo orden

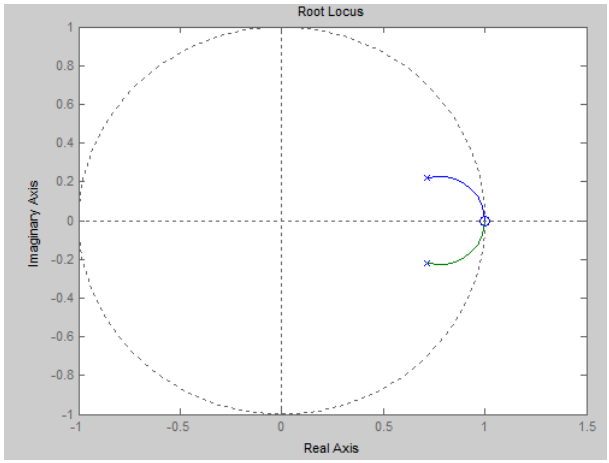

Figura 53. Diagrama de polos y ceros del filtro Butterworth paso alto de segundo orden.

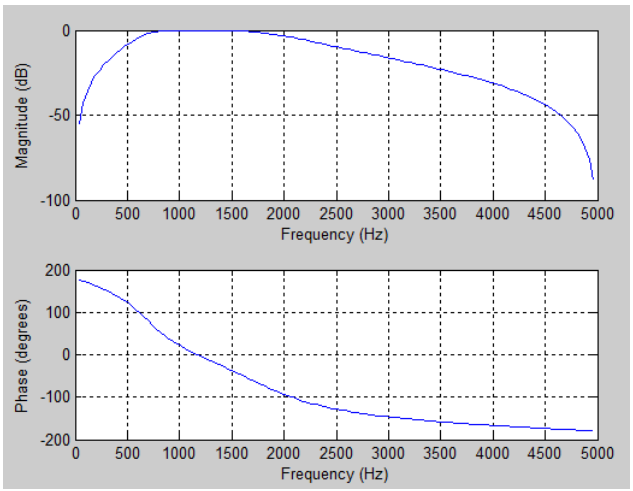

Figura 54. en decibeles y en fase de un filtro Butterworth paso banda de segundo orden

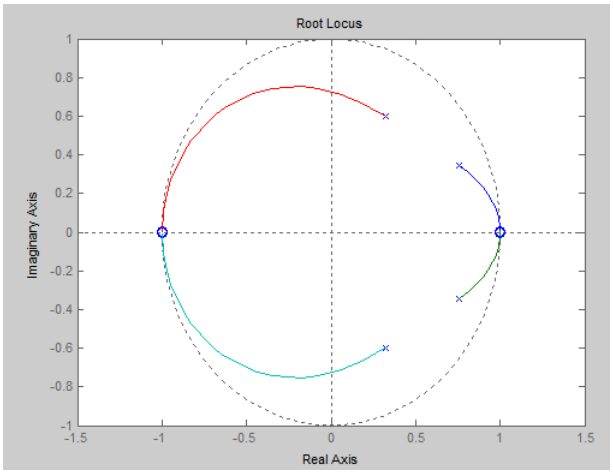

Figura 55. Diagrama de polos y ceros del filtro Butterworth paso banda de segundo orden.

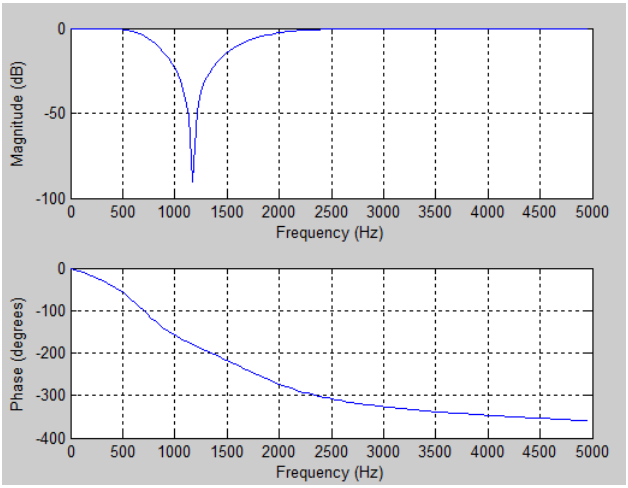

Figura 56. en decibeles y en fase de un filtro Butterworth paso banda eliminada de segundo orden

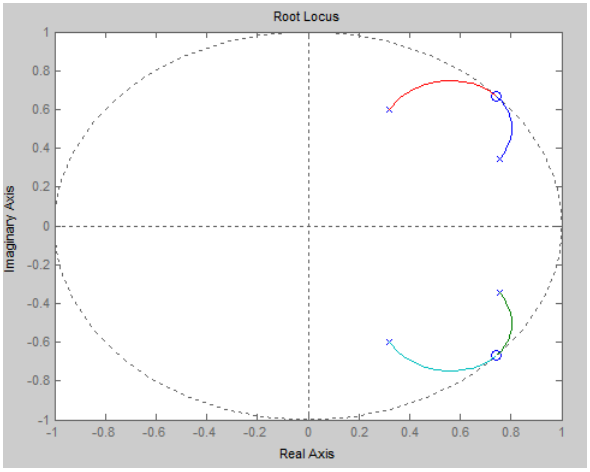

Figura 57. Diagrama de polos y ceros del filtro Butterworth paso banda eliminada de segundo orden.

Como se puede observar en las figuras anteriores, los comportamientos son los esperados para cada uno de todos los tipos de filtros, paso bajo, paso alto, paso banda y paso banda eliminada.

6.2. FILTRO CHEBYSHEV

6.2.1. FILTRO CHEBYSHEV TIPO 1

Para el desarrollo de este filtro se utilizó el siguiente comando en Matlab, [B,A]=cheby1(N,R,Wn). En el Anexo A, se puede observar los comandos utilizados para la implementación y estudio del filtro Chebyshev del tipo 1.

Como se podrá observar, en las siguientes figuras se podrá ver el comportamiento que tendrán los filtros Chebyshev de tipo 1 para la implementación de filtros paso bajo, paso alto, paso banda y paso banda eliminada.

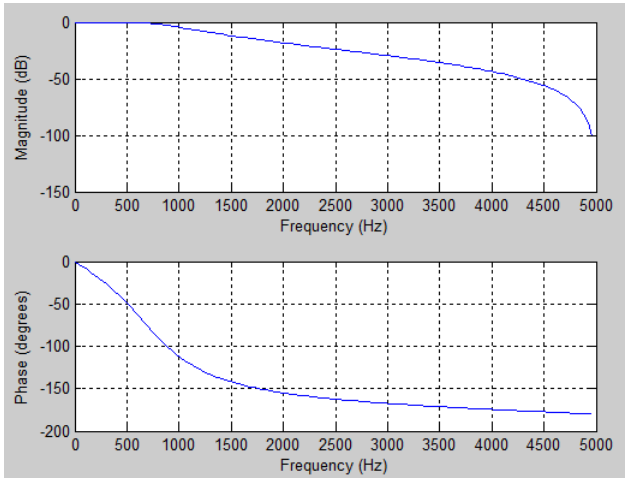

Figura 58. Grafica en decibeles y en fase de un filtro Chebyshev tipo 1 paso bajo de segundo orden

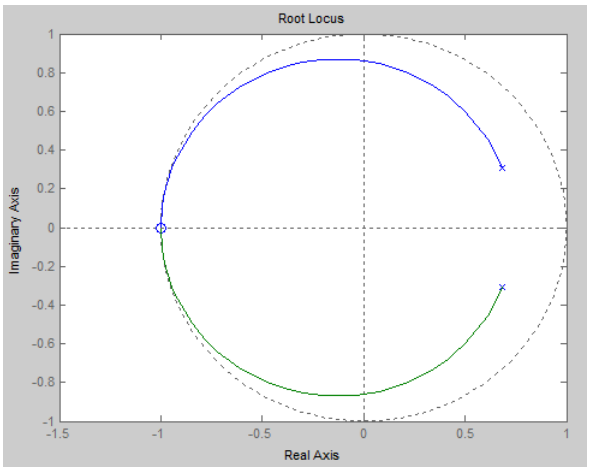

Figura 59. Diagrama de polos y ceros del filtro Chebyshev tipo 1 paso bajo de segundo orden.

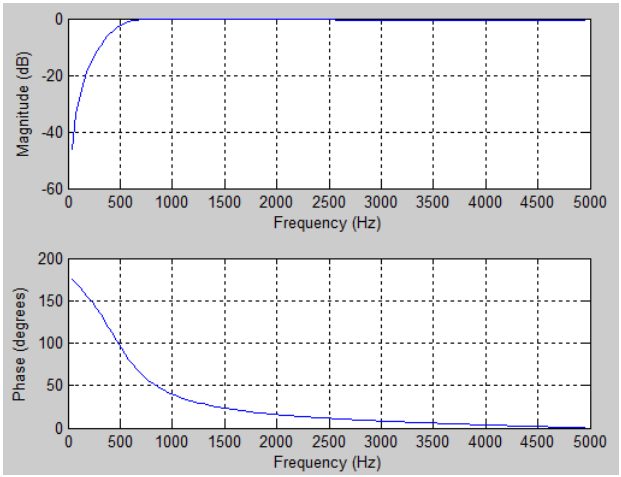

Figura 60. en decibeles y en fase de un filtro Chebyshev tipo 1 paso alto de segundo orden

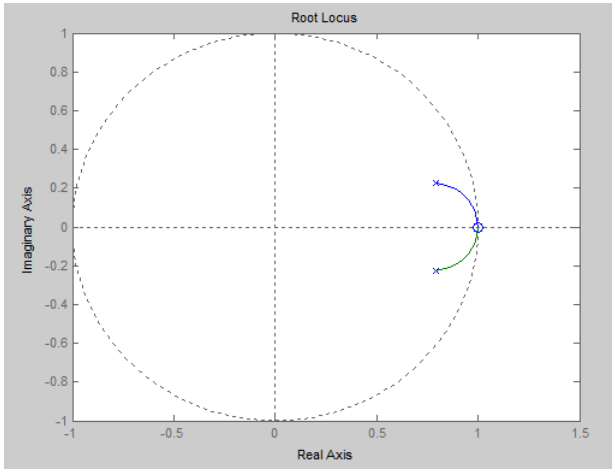

Figura 61. Diagrama de polos y ceros del filtro Chebyshev tipo 1 paso alto de segundo orden.

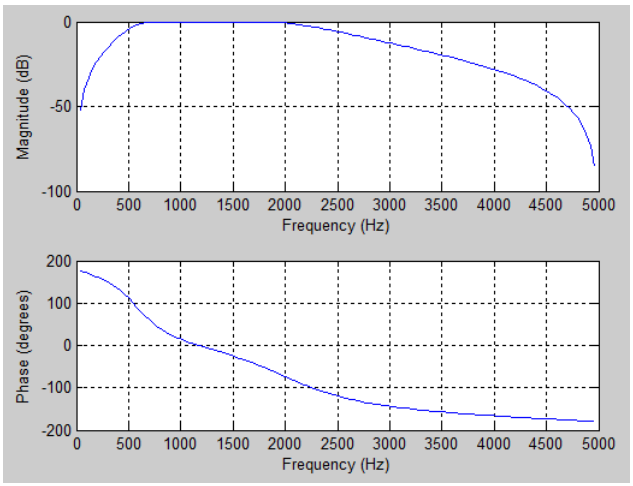

Figura 62. en decibeles y en fase de un filtro Chebyshev tipo 1 paso banda de segundo orden

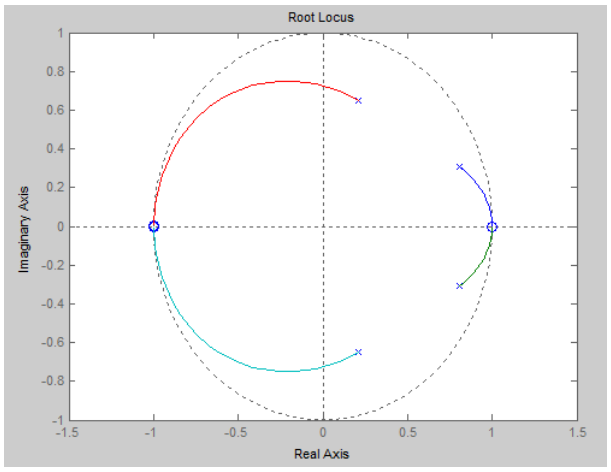

Figura 63. Diagrama de polos y ceros del filtro Chebyshev tipo 1 paso banda de segundo orden.

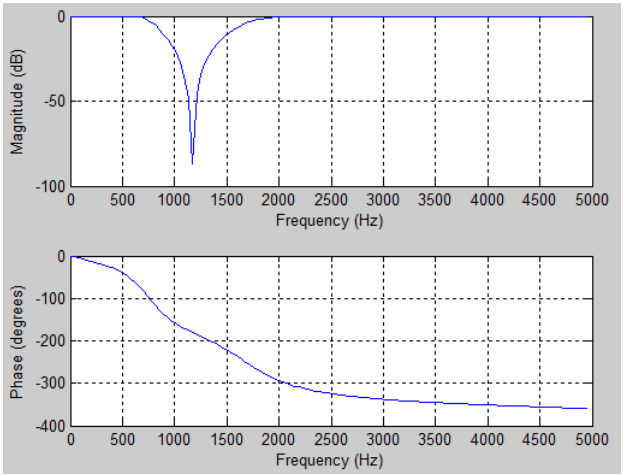

Figura 64. en decibeles y en fase de un filtro Chebyshev tipo 1 paso banda eliminada de segundo orden

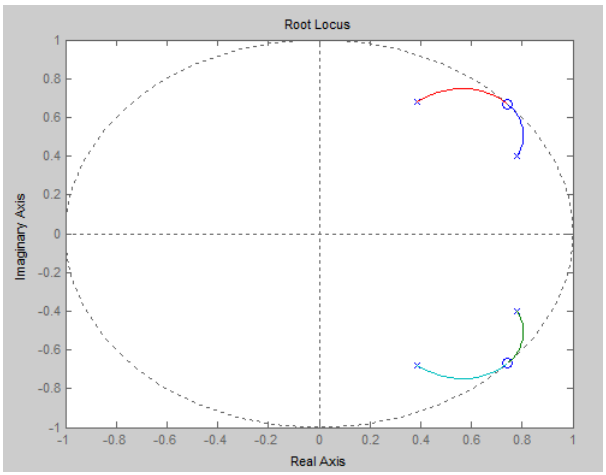

Figura 65. Diagrama de polos y ceros del filtro Chebyshev tipo 1 paso banda eliminada de segundo orden.

Como se puede observar en las figuras anteriores, los comportamientos son los esperados para cada uno de todos los tipos de filtros, paso bajo, paso alto, paso banda y paso banda eliminada.

6.2.2. FILTRO CHEBYSHEV TIPO 2

Para el desarrollo de este filtro se utilizó el siguiente comando en Matlab, [B,A]=cheby2(N,R,Wn). En el Anexo A, se puede observar los comandos utilizados para la implementación y estudio del filtro Chebyshev del tipo 2.

Como se podrá observar, en las siguientes figuras se podrá ver el comportamiento que tendrán los filtros Chebyshev de tipo 2 para la implementación de filtros paso bajo, paso alto, paso banda y paso banda eliminada.

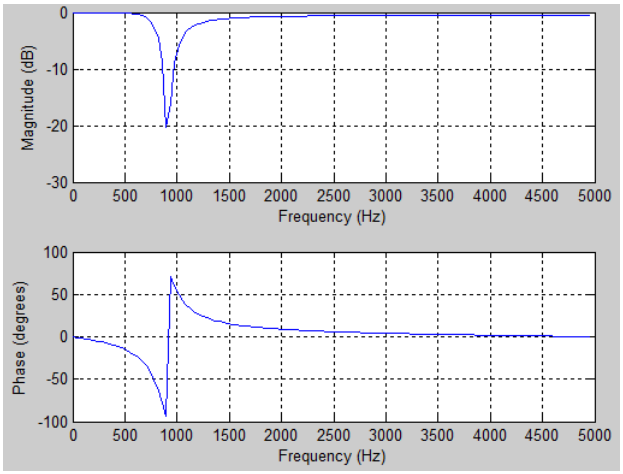

Figura 66. Grafica en decibeles y en fase de un filtro Chebyshev tipo 2 paso bajo de segundo orden

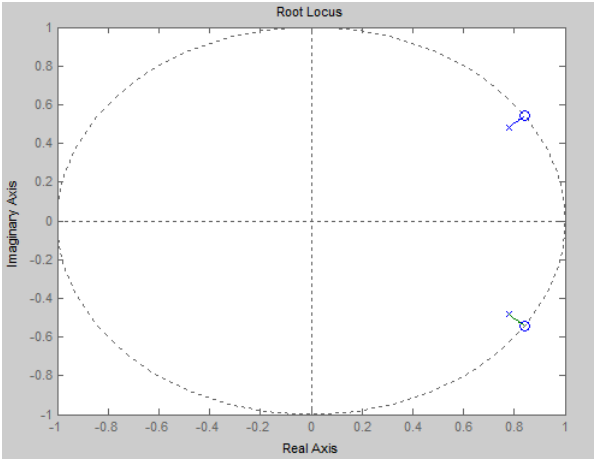

Figura 67. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso bajo de segundo orden.

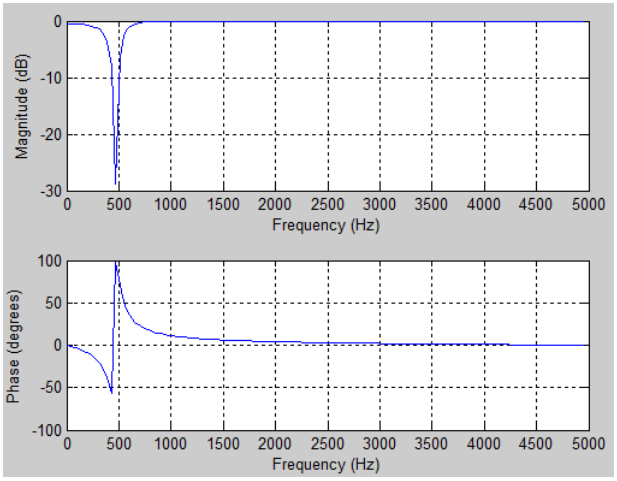

Figura 68. en decibeles y en fase de un filtro Chebyshev tipo 2 paso alto de segundo orden

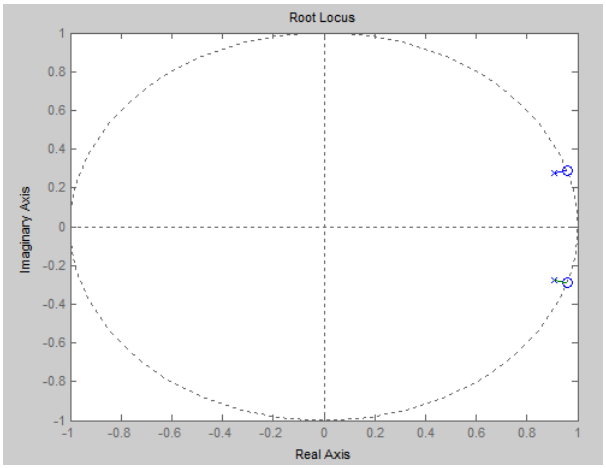

Figura 69. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso alto de segundo orden.

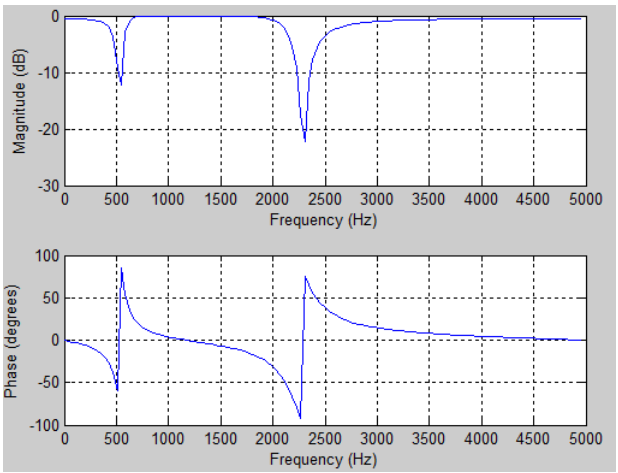

Figura 70. en decibeles y en fase de un filtro Chebyshev tipo 2 paso banda de segundo orden

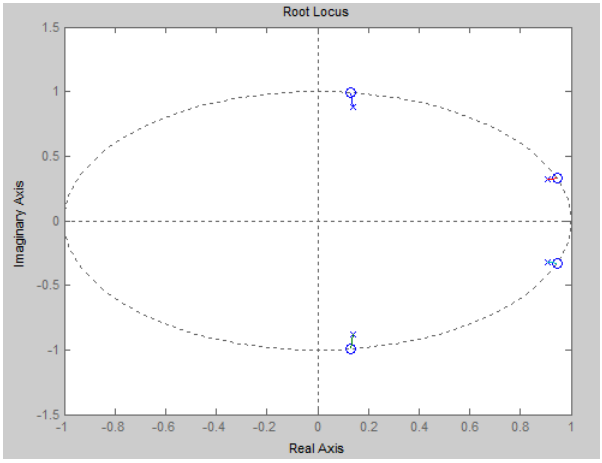

Figura 71. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso banda de segundo orden.

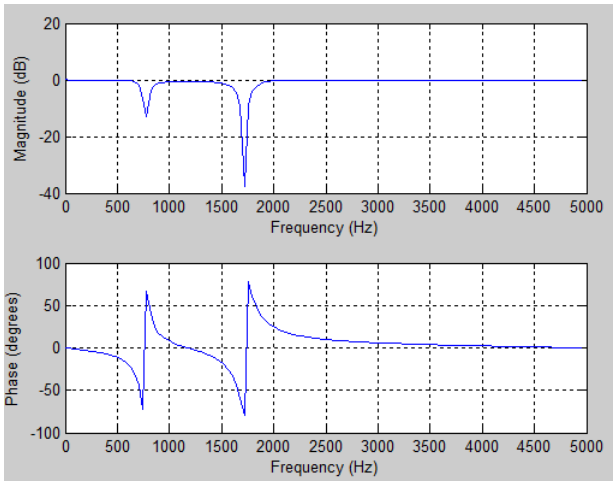

Figura 72. en decibeles y en fase de un filtro Chebyshev tipo 2 paso banda eliminada de segundo orden

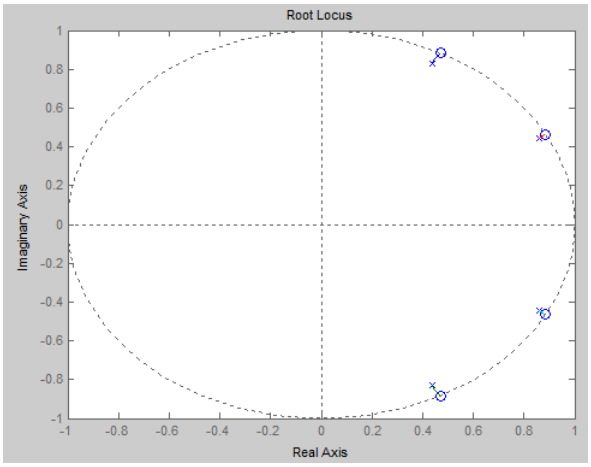

Figura 73. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso banda eliminada de segundo orden.

Como se puede observar en las figuras anteriores, los ceros aumentan en la ventana de tal forma que se atenúa más, perdiendo así los risos del filtro y denotando más los polos, de igual manera se puede ver cambios muy herráticos en la respuesta de fase del filtro

6.3. FILTRO ELÍPTICO O DE CAUER

Para el desarrollo de este filtro se utilizó el siguiente comando en Matlab, [B,A]=ellip(N,R,Wn). En el Anexo A, se puede observar los comandos utilizados para la implementación y estudio del filtro Chebyshev del tipo 2.

Como se podrá observar, en las siguientes figuras se podrá ver el comportamiento que tendrán los filtros Chebyshev de tipo 2 para la implementación de filtros paso bajo, paso alto, paso banda y paso banda eliminada.

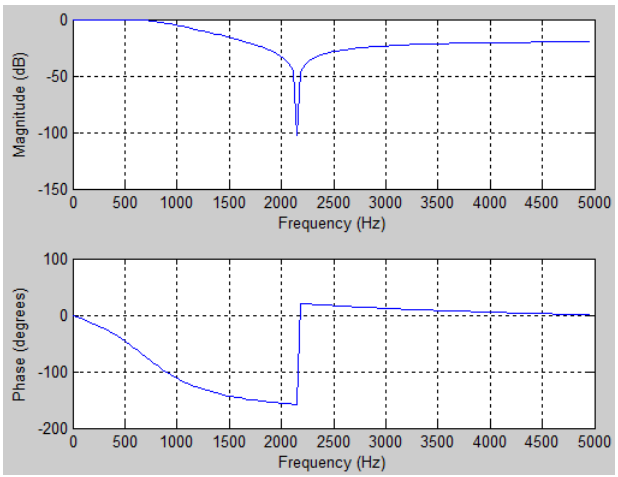

Figura 74. Grafica en decibeles y en fase de un filtro Chebyshev tipo 2 paso bajo de segundo orden

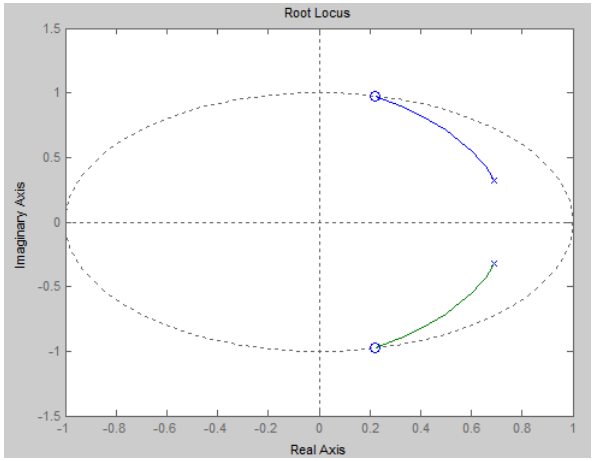

Figura 75. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso bajo de segundo orden.

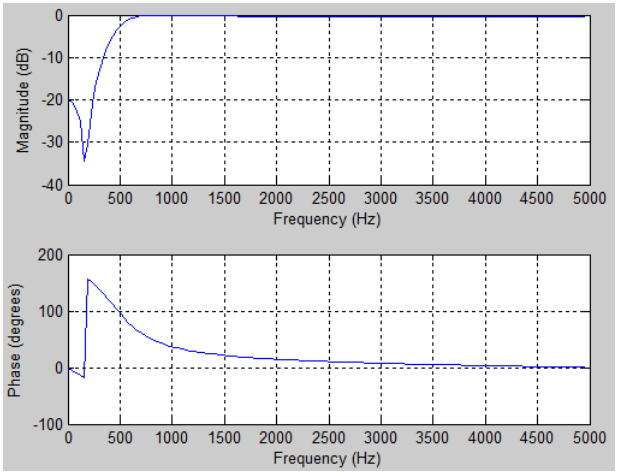

Figura 76. en decibeles y en fase de un filtro Chebyshev tipo 2 paso alto de segundo orden

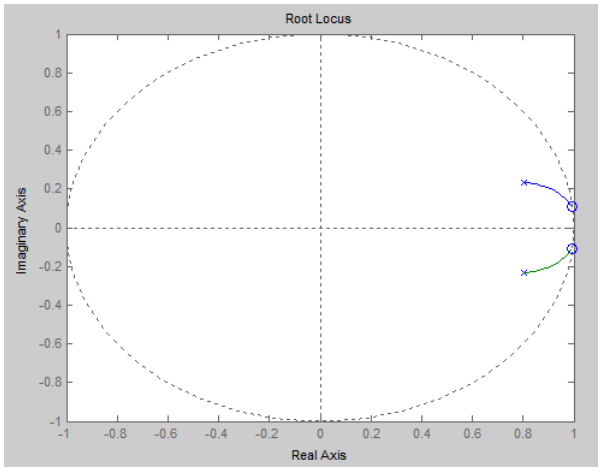

Figura 77. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso alto de segundo orden.

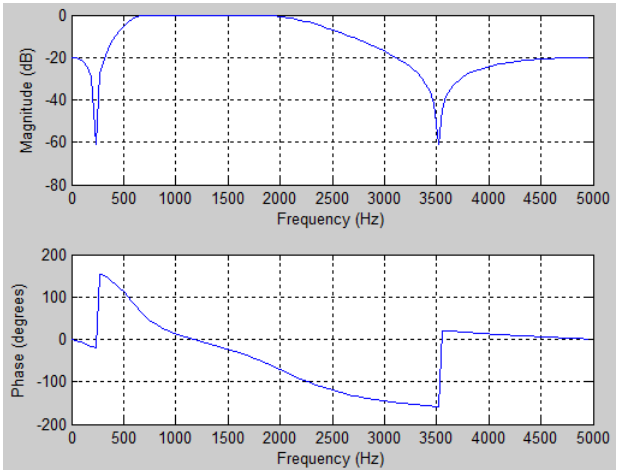

Figura 78. en decibeles y en fase de un filtro Chebyshev tipo 2 paso banda de segundo orden

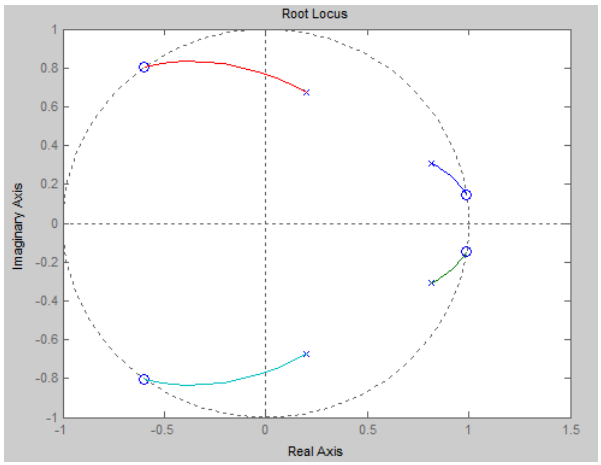

Figura 79. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso banda de segundo orden.

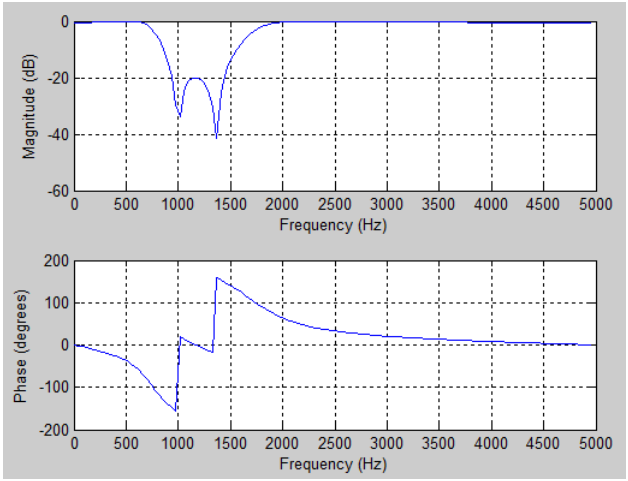

Figura 80. en decibeles y en fase de un filtro Chebyshev tipo 2 paso banda eliminada de segundo orden

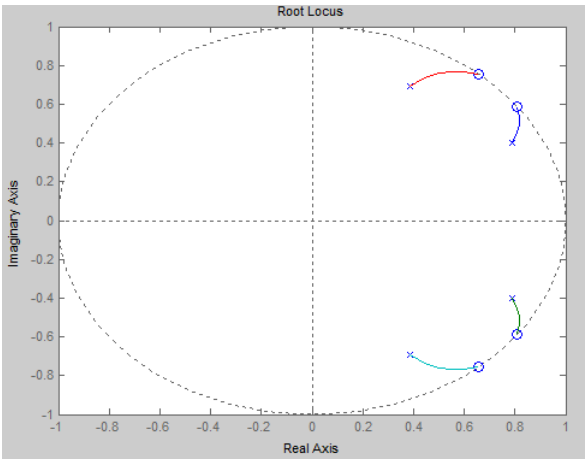

Figura 81. Diagrama de polos y ceros del filtro Chebyshev tipo 2 paso banda eliminada de segundo orden.

Como se puede observar en las figuras anteriores, los cortes son más firmes llegando casi a semejarse a filtros ideales.

6.4. FILTRO ADAPTATIVO LMS

Como bien se comento en párrafos anteriores, este proyecto se centró en la ejecución de los filtros adaptativos, en especial los filtros adaptativos LMS (Least Mean Square); debido a esto, en esta parte, se mostrará resultados a la salida del DSP, como también a la salida de la tarjeta de pruebas; de entradas, que fueron excitadas por el Cool Edit Pro y por el acelerómetro.

Dicho esto primero observemos el comportamiento de nuestro filtro adaptativo LMS en Matlab:

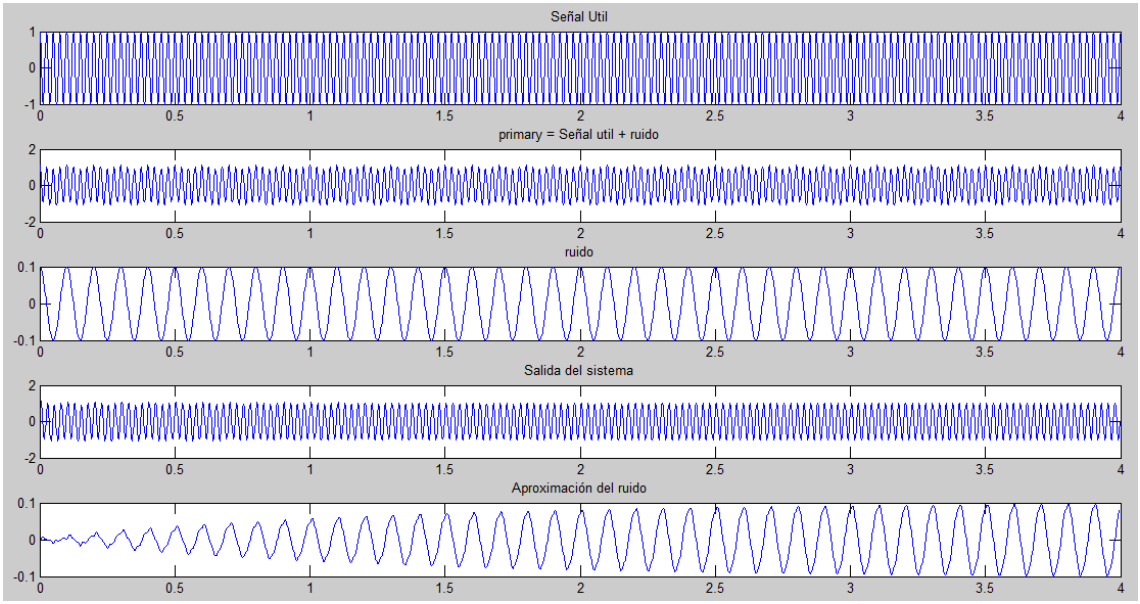

Figura 82. Identificación de señales del filtro LMS que deseamos implementar.

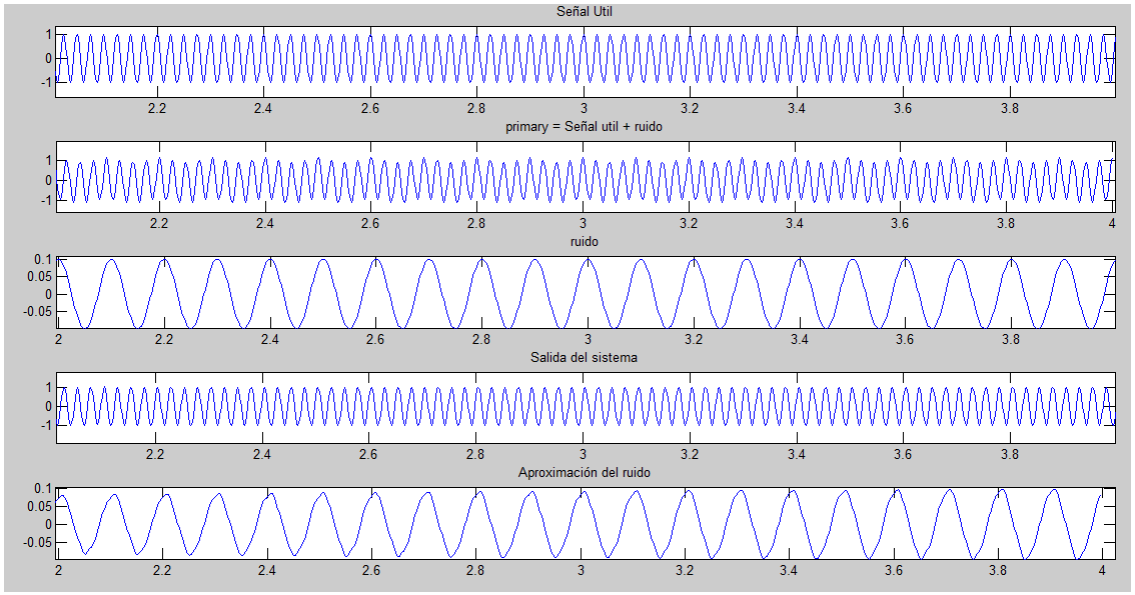

Figura 83. Realizacion de Zoom sobre las figuras anteriores.

Como bien podemos observar la aproximación de ruido y la señal de ruido, son casi parecidas a partir del instante dos, ergo nuestra señal de salida del sistema se asemeja bastante a nuestra señal útil.

Resaltaremos además, que para la realización de esta prueba se utilizó dos señales escaladas, nuestra señal útil es de cuatro kilo Herzios, nuestra señal de ruido de un kilo Herzio, y nuestra frecuencia de muestreo es de cuarenta y ocho kilo Herzios. Tal como hemos

mencionado, las señales están escaladas, para tener una visión mejor y ampliada de su funcionamiento, es decir cada frecuencia está dividida entre cien.

7. DESCRIPCIÓN EXPERIMENTAL CON EL DSP

7.1. PRUEBAS CON LOS FILTROS DIGITALES

Tras el estudio efectuado con los filtros digitales; Butterworth, Chebyshev, y elíptico; en sus distintos tipos; paso bajo, paso alto, paso banda eliminada y paso banda; en el entorno de Matlab, observar el comportamiento que debería tener, y la forma de cómo implementarlo, pasaremos a "cargarlo", en nuestro DSP, para realizar las pruebas pertinentes.

Para la implementación de estos filtros, hizo falta saber la ecuación en diferencias, el cual se puede sacar mediante Matlab, gracias al siguiente comando:

```
H=tf(B,A,Tm);
```

Quien, como se explico en anteriores apartados, nos permite obtener la función de transferencia a partir de ciertos parámetros solicitados por esta función (coeficientes de numerador, del denominador y el tiempo de muestreo). Con la función de transferencia obtenida, la ecuación en diferencias se puede deducir de ella.

Dicho lo cual podemos observar un modelo del código implementado en el DSP, para los filtros digitales, en el anexo x. En donde se observa cómo se llegó a utilizar las ecuaciones en diferencias, y cómo pasamos al filtrado con este tipo de filtros.

El código es aparentemente sencillo, para cada filtro utilizado, ya sea Butterworth, Chebyshev, y elíptico; o paso bajo, paso alto, paso banda eliminada y paso banda; puesto que se centra exclusivamente en, lo que ya hemos acarreado y dado énfasis desde el principio de este apartado, la ecuación en diferencias. Como se podrá deducir, a mayor orden del filtro, mas coeficientes tendrá la ecuación en diferencias, por lo tanto más larga será la línea a implementar para el filtro, y por supuesto más variables habrá que definir.

En las siguientes figuras, observaremos algunas de las respuestas de los filtros digitales, luego de implementarlas en el DSP.

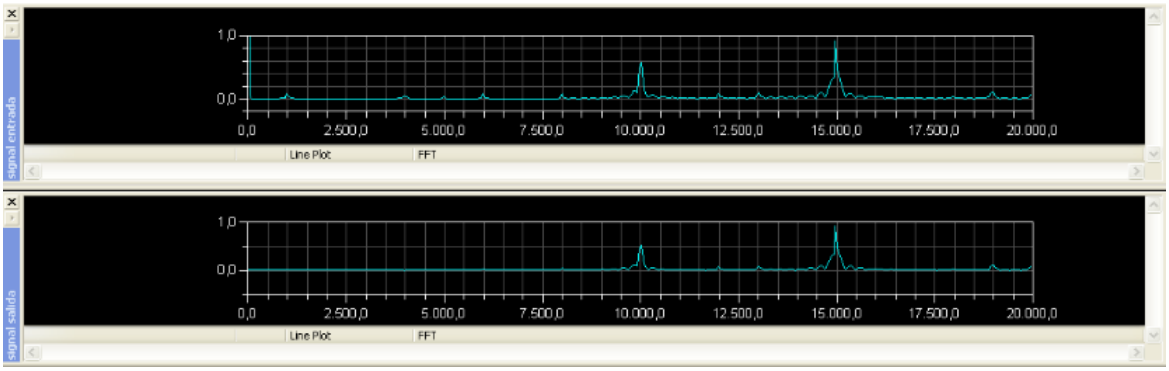

Figura 84. Señales de entrada y salida tras implementar filtro de orden 3 de Butterworth paso alto, con frecuencia de corte en 8kHz.

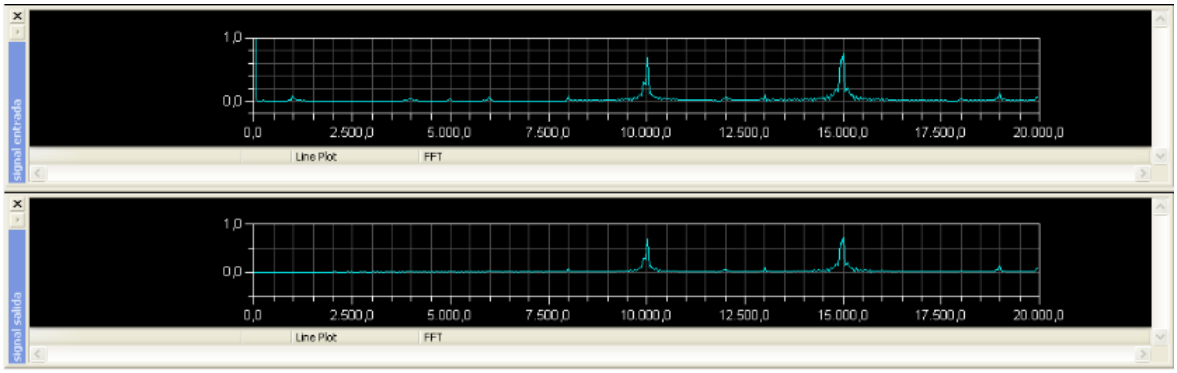

Figura 85. Señales de entrada y salida tras implementar filtro de orden 3 de Chebyshev de tipo 1 paso alto, con frecuencia de corte en 8kHz.

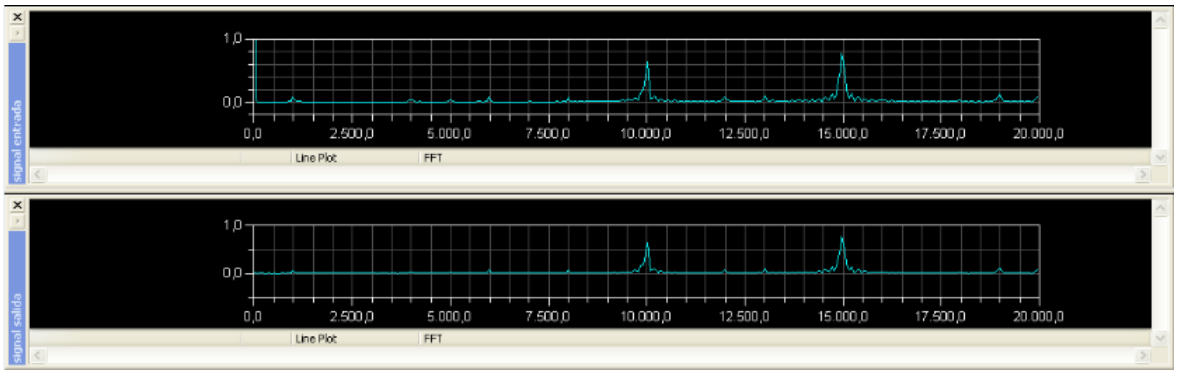

Figura 86. Señales de entrada y salida tras implementar filtro de orden 3 de Chebyshev de tipo 2 paso alto, con frecuencia de corte en 8kHz.

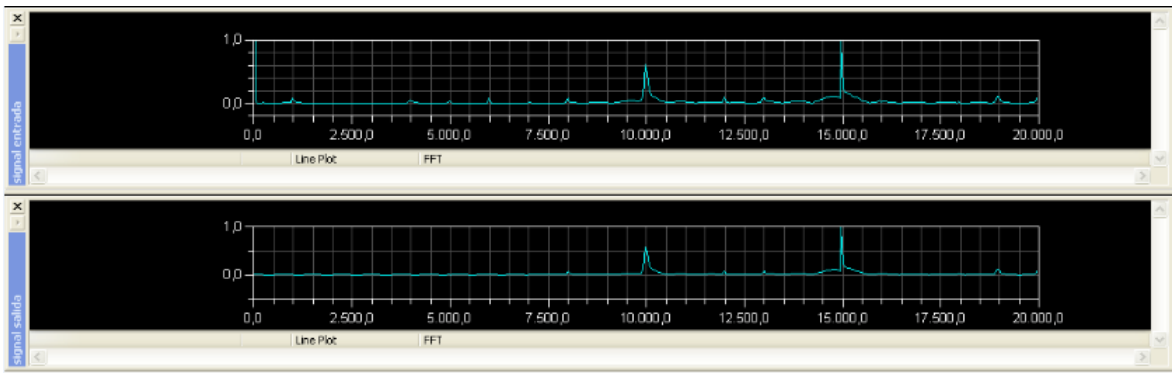

Figura 87. Señales de entrada y salida tras implementar filtro de orden 3 Elíptico paso alto, con frecuencia de corte en 8kHz.

Como podemos observar el comportamiento para cada uno de los filtros es el esperado, resaltando que a pesar que la frecuencia de corte es en 8kHz, existe una atenuación en la señal de 10kHz, y las señales próximas a 8kHz también se ven atenuadas, esto es debido a que la respuesta en frecuencia no es ideal, es decir la banda de transición no es inmediata, sino que cae muy lento. Se podría solucionar aumentado el orden del filtro, pero esto conlleva a lo que expresábamos con anterioridad, introducir más coeficientes en la ecuación en diferencias, introducir más coeficientes y además hacer más extensa la línea de comando del filtro digital a implementar.

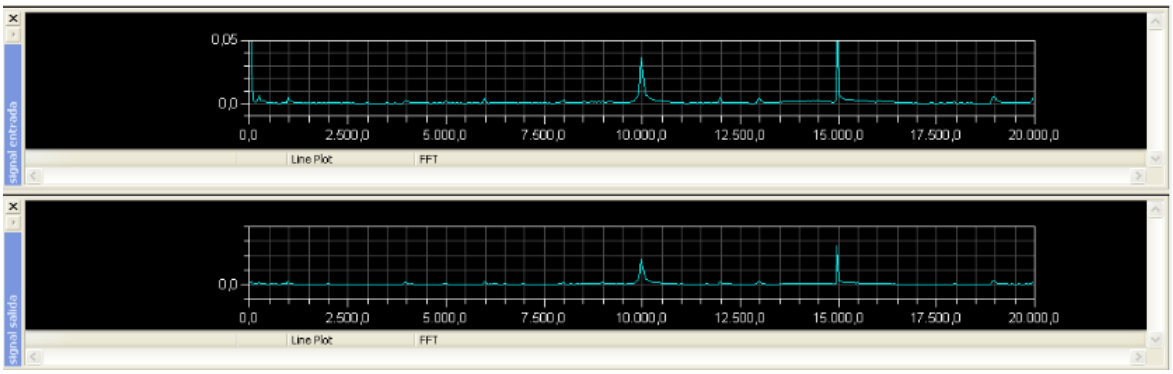

Figura 88. Señales de entrada y salida tras implementar filtro de orden 5 Chebyshev de tipo 2 paso banda, con frecuencia de corte en 5kHz y 12kHz.

En la figura anterior, se puede observar con mayor claridad este efecto, observamos que ambas señales, que llamaremos útiles, de 10kHz y 15kHz, se ven atenuadas por el filtro paso banda implementado de orden 5.

Con esto ya ejecutado y analizado, pasaremos a la implementación de nuestro filtro adaptativo LMS.

7.2. PRUEBAS CON FILTROS LMS

Tras haber estudiado el comportamiento de los filtros digitales y de observar que nuestro algoritmo LMS funciona como se esperaba en Matlab, procederemos a realizar una serie de pruebas, experimentos; en el DSP, integrado en el kit de desarrollo ADSP-21161 EZ-KIT, excitando primero con señales provenientes del Cool Edit Pro++, y terminando con señales provenientes directamente del acelerómetro.

A continuación pasaremos a observar el comportamiento de nuestro filtro LMS ante señales provocadas por el Cool Edit Pro:

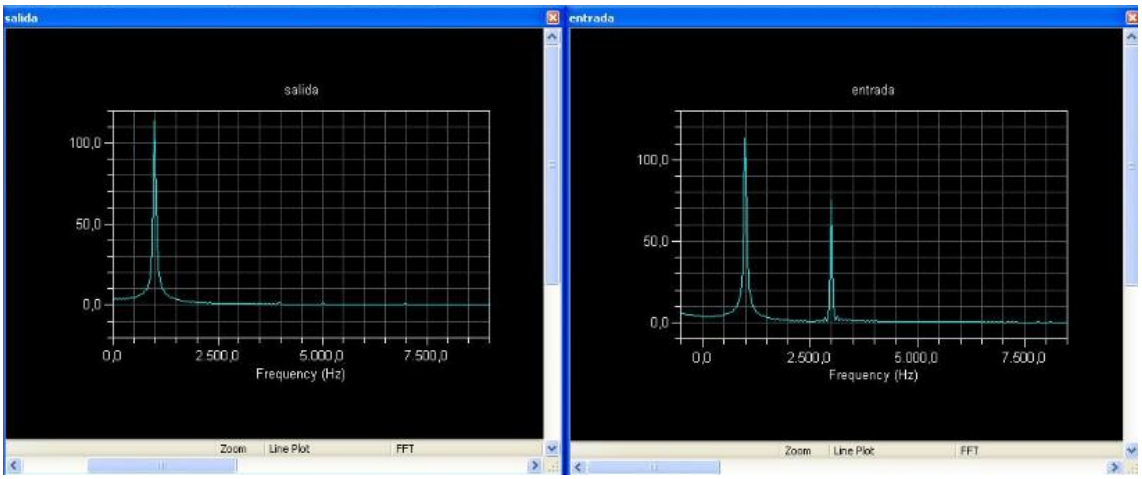

Figura 89. Respuesta de nuestro filtro LMS frente a señal de entrada de 1kHz y 3kHz; y 3kHz como ruido.

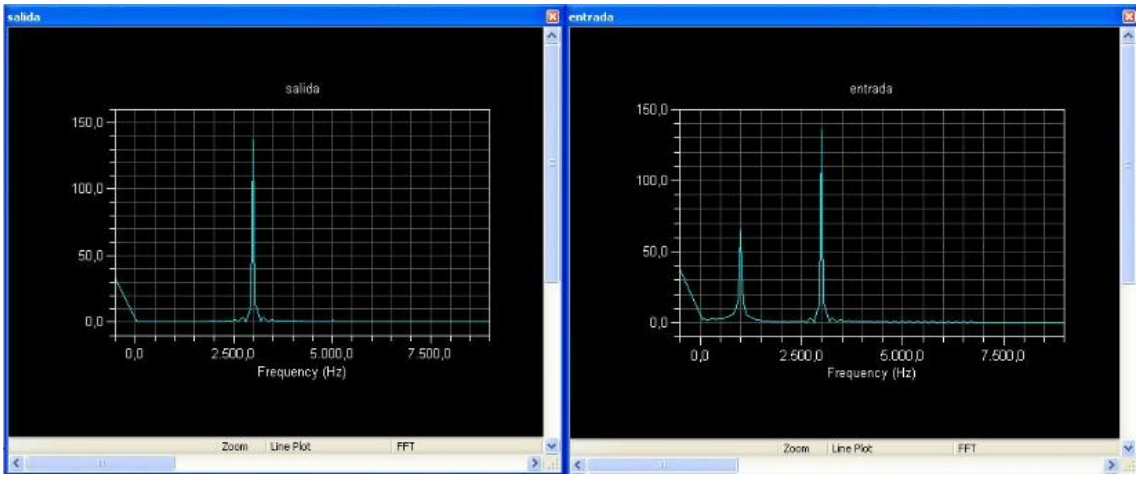

Figura 90. Respuesta de nuestro filtro LMS frente a señal de entrada de 3kHz y 1kHz; y 1kHz como ruido.

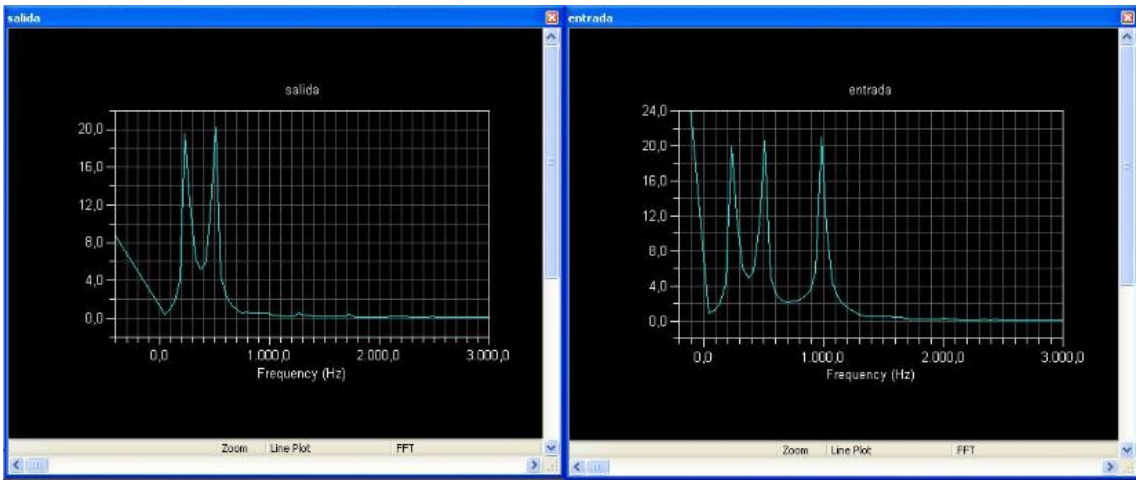

Figura 91. Respuesta de nuestro filtro LMS frente a señal de entrada de 200Hz, 500Hz y 1kHz; y 1kHz como ruido.

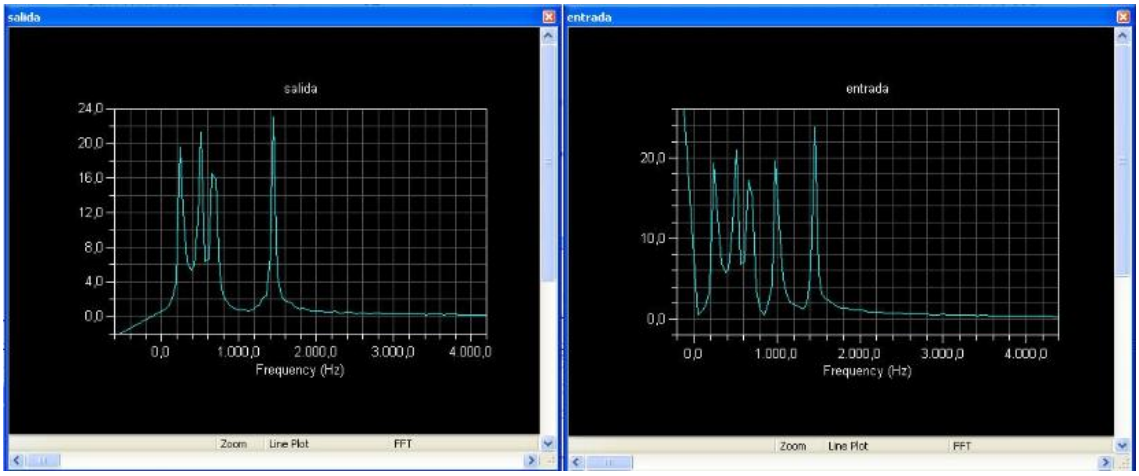

Figura 92. Respuesta de nuestro filtro LMS frente a señal de entrada de 200Hz, 500Hz, 800Hz 1kHz y 1.2kHz; y 1kHz como ruido.

Como se puede observar en las graficas previas, las señale que deseamos eliminar (señales de ruido), se atenúan por completo, en cualquier ubicación que se encuentre la frecuencia a filtrar.

Resaltar que los dibujos mostrados, son los resultados de la herramienta Visual DSP++, tanto para la entrada como para la salida.

Además, los filtros implementados, siguen los algoritmos estudiados con anterioridad, manteniendo un orden del filtro de seis; la señal de entrada es provocada por el Cool Edit Pro , como bien se mencionó, provocando excitaciones a la entrada de la tarjeta de prueba que posteriormente pasaran al DSP, que procesará las señales y filtrará, de acuerdo a los parámetros determinados, las señales no deseadas, produciendo la salida que se observa en las graficas anteriores.

Cuando se tiene las señales de salida justo a la salida del DSP, este pasará a los canales de salida de la tarjeta, captando dichas señales con ayuda de un osciloscopio:

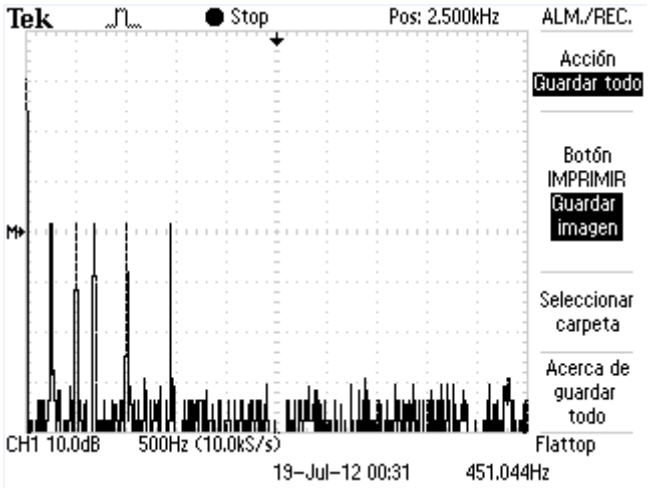

Figura 93. Señal de entrada a la tarjeta de pruebas.

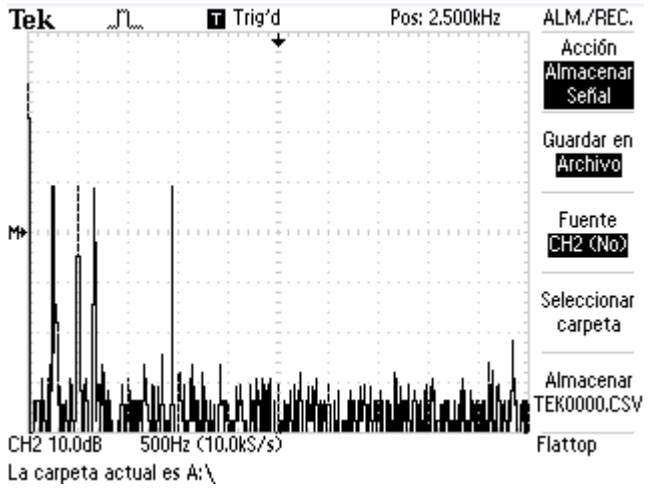

Figura 94. Señal de salida a la tarjeta de pruebas, con señal de 1kHz filtrada.

Con esto observamos que el filtro adaptativo LMS, implementado funciona también justo y después de la salida de la tarjeta de pruebas, resaltando además que la señal filtrada es prácticamente atenuada por completo.

Con estos resultados, pasaremos a comprobar nuestro filtro adaptativo LMS, frente a señales desfasadas. Para ello será necesario elevar el orden del filtro a doscientos, para cubrir toda la gama de frecuencias y mantener una estabilidad de filtrado en la salida para todas estas señales.

Dicho esto, en las siguientes imágenes, se pueden observar dichas pruebas, resaltar que las señales de entrada se representaran en color verde, y las señales de salida en violeta:

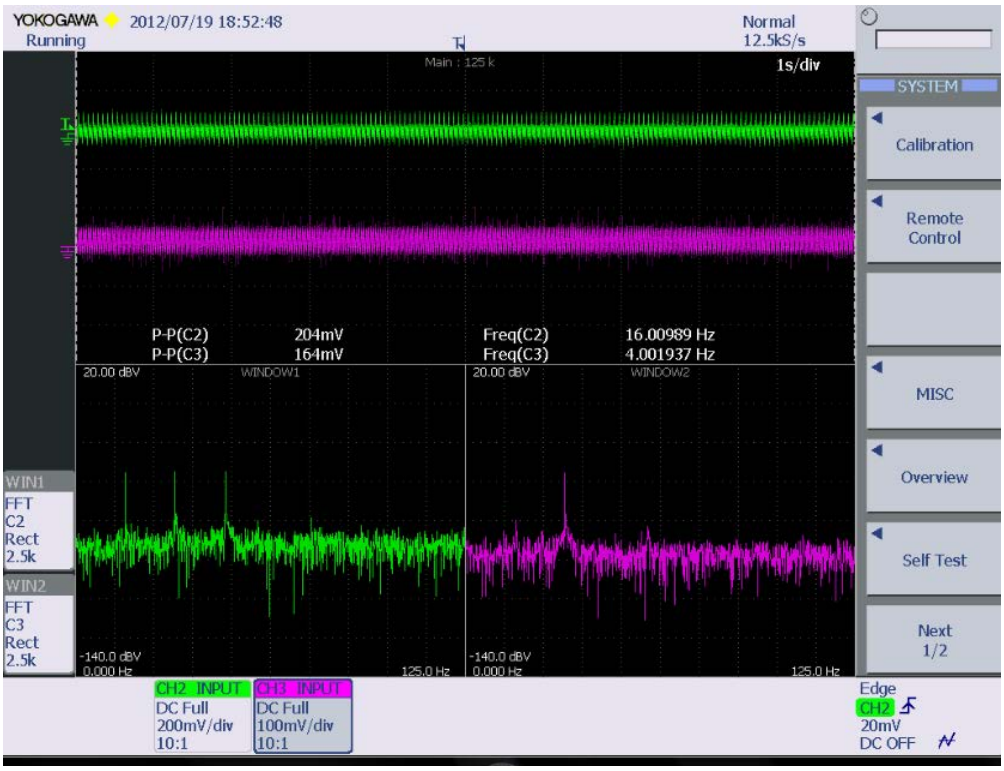

Figura 95. Señal de entrada de 16Hz, 32Hz y 48Hz, y salida filtrada

Se puede observar que se han filtrado dos señales, la señal de 16Hz y la de 48Hz, dejando de esta manera la señal de 32Hz en la salida.

Cabe resaltar, como se dijo con anterioridad, que la señal de entrada y la señal de referencia están desfasadas.

Con esta parte ya realizada, ejecutada y analizada, procederemos a ir al laboratorio, en donde se pasará al análisis de nuestro algoritmo, frente a señales provenientes del acelerómetro 4000A.

Para ello, os presentamos en las siguientes figuras, el entorno en el que se trabajó esta última etapa, y en donde se realizaron las pertinentes pruebas:

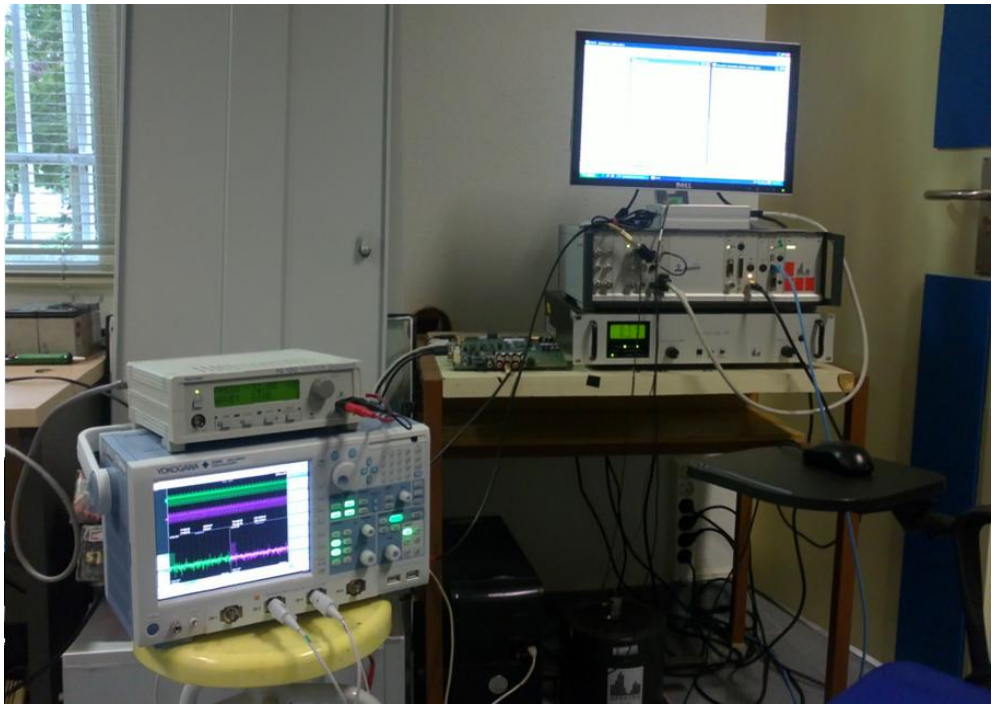

Figura 96. Laboratorio de calibración de Acelerómetros.

Figura 97. Acelerómetro sobre el Shaker.

Con las pruebas realizadas en Matlab, y las pruebas ejecutas sobre el DSP, con señales provenientes del Cool Edit Pro, estamos listos para pasar a probar nuestro filtro adaptativo LMS, con señales reales, es decir, con señales provenientes del acelerómetro.

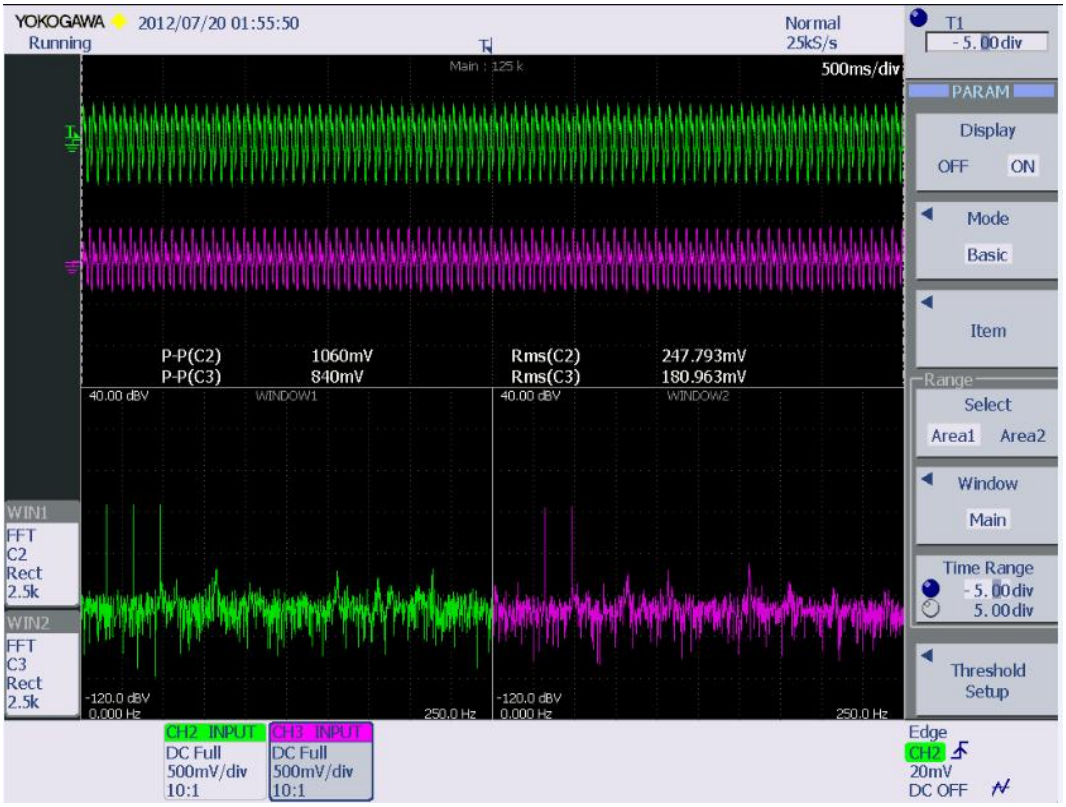

Figura 98. Señal de entrada 16Hz, 32Hz y 48Hz. Señal de salida con 16Hz filtrada.

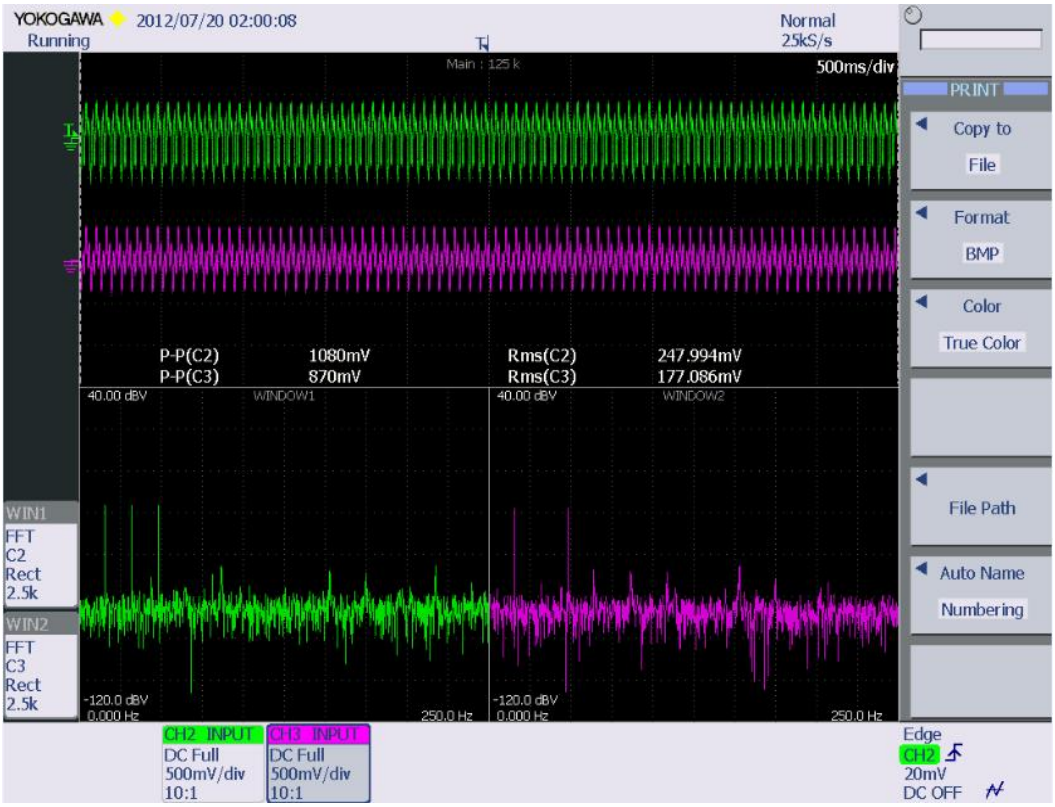

Figura 99. Señal de entrada 16Hz, 32Hz y 48Hz. Señal de salida con 32Hz filtrada.

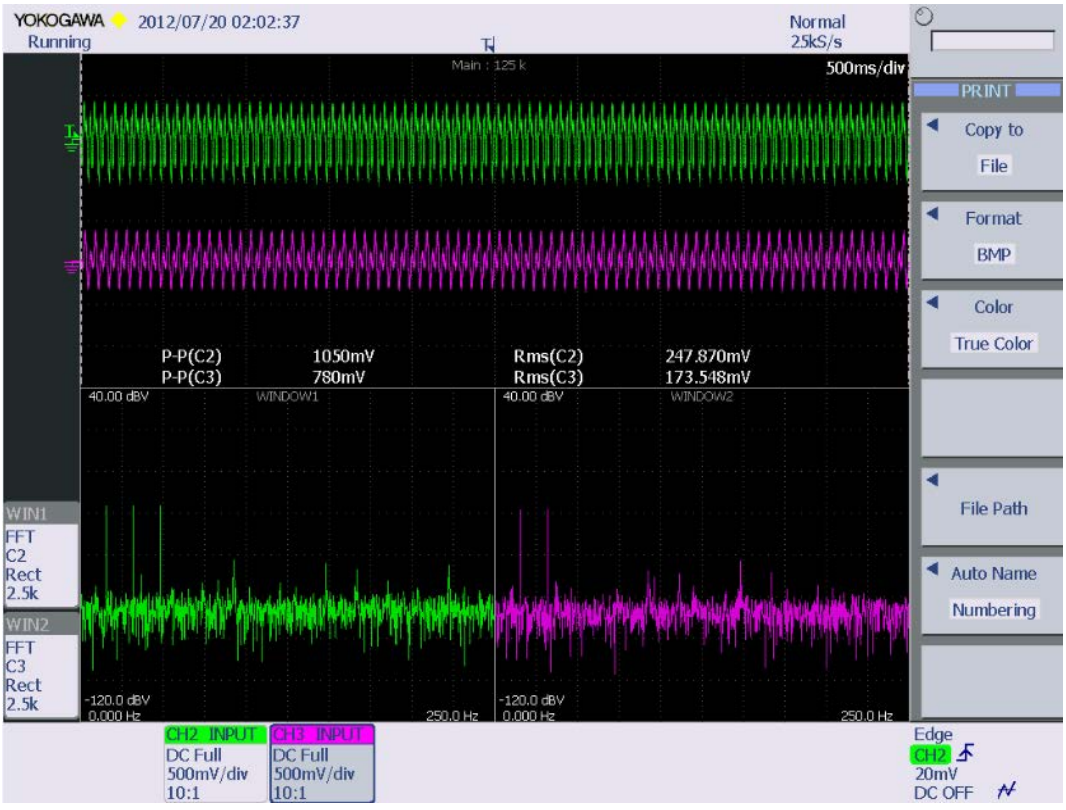

Figura 99. Señal de entrada 16Hz, 32Hz y 48Hz. Señal de salida con 48Hz filtrada.

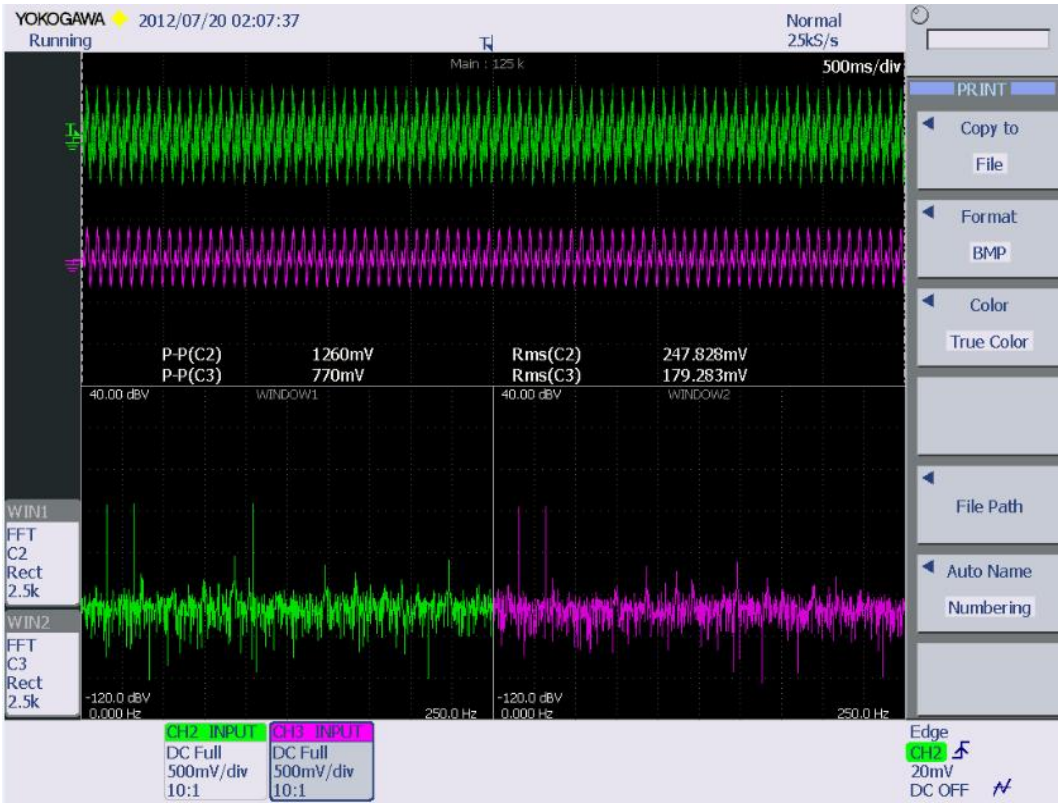

Figura 100. Señal de entrada 16Hz, 32Hz y 100Hz. Señal de salida con 100Hz filtrada.

Figura 101. Señal de entrada 16Hz, 32Hz y 100Hz. Señal de salida con 16Hz filtrada.

Figura 102. Señal de entrada 16Hz, 32Hz y 100Hz. Señal de salida con 32Hz filtrada.

En las imágenes anteriores, se puede observar, que las pruebas se realizaron con señales con frecuencias bajas, muy cercanas y lejanas, obteniendo resultados esperados y con atenuaciones de las señales filtradas prácticamente por completo.

Resaltar además, que la señal de referencia, es decir la señal que provocará la eliminación de una de las tres señales de entrada, se provoco con la ayuda de un generador de señales, el cual implementaba un desfase frente a la señal de entrada. Recordar además que debido al desfase involucrado, tuvimos que aumentar el orden del filtro (orden igual a doscientos), para cubrir todas las frecuencias y mantener la estabilidad de la señal de salida y eliminar por completo la señal no deseada, como bien se puede apreciar en los resultados capturados por el osciloscopio.

Finalmente recalcar, que en el anexo x, se encuentra el código implementado en el DSP para el filtro adaptativo LMS, comentado brevemente en cada línea.

7.3. PRUEBAS CON FILTROS PREDICTIVOS

A modo de actividad adicional, se estudio el filtro predictivo, implementándolo en el DSP, y excitamos la entrada del sistema con distintas señales, produciendo, en efecto, los comportamientos que pasaremos a detallar en breve.

En las siguientes figuras, se puede observar algunas de las pruebas que realizamos tras excitar nuestro sistema, con el filtro predictivo implementado:

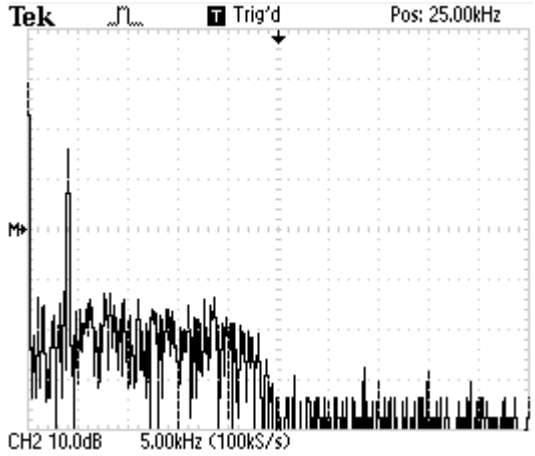

Figura 103. Entrada del sistema, señal de 4kHz con ruido blanco

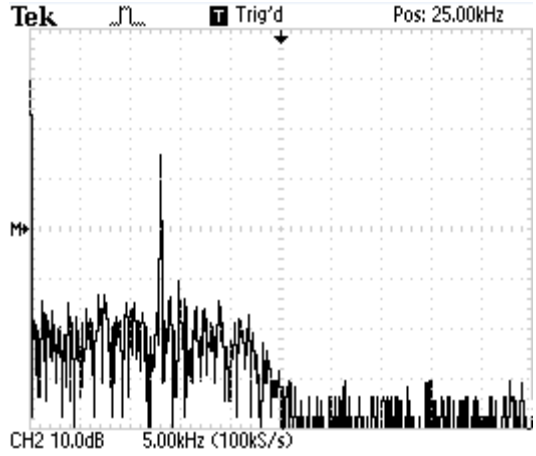

Figura 104. Entrada del sistema, señal de 12kHz con ruido blanco

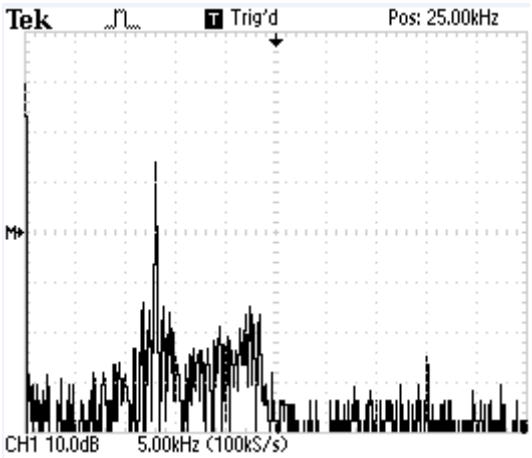

Figura 105. Salida del sistema, señal de 12kHz con ruido blanco filtrado

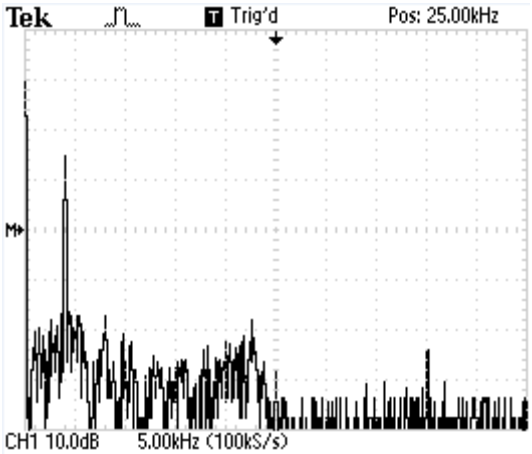

Figura 106. Salida del sistema, señal de 4kHz con ruido blanco filtrado

Como se puede observar, se utilizó para este caso, dos señales de entrada, una señal de 4kHz de frecuencia y otra de 12kHz.

Con estas dos señales “útiles”, se adjunto también ruido blanco, que tras realizar el filtrado predictivo, podemos observar como este logra eliminar gran parte de este ruido.

Rescatamos también de las imágenes, que a las salidas se aprecia unas protuberancias en forma de “montañas”, después de la señal útil hasta el final de aparición del ruido blanco. Dichas protuberancias, se van haciendo más angostas, conforme se aumenta el orden del filtro adaptativo. Además tras aumentar el orden del filtro, las protuberancias aumentan, pero disminuyen en amplitud.

Como el objetivo del presente proyecto eran los filtros adaptativos LMS, y no los filtros predictivos, no se emplea mayor énfasis en su elaboración y estudio.

Por último resaltar que en el anexo X tenemos nuestro código implementado para el filtro predictivo, el cual presenta unos breves comentarios.

8. APLICACIONES DE LOS FILTROS ADAPTATIVOS Y LOS FILTROS ADAPTATIVOS LMS

Los filtros adaptativos, en todas sus clases, presentan distintas aplicaciones en diferentes campos.

Según artículos obtenidos por la base de datos de la EUITT, podemos destacar cuatro de ellos, en donde nos relatan algunos de sus multiples usos de estos filtros, los filtros adaptativos.

Destacaremos y haremos mayor énfasis en las aplicaciones de los filtros adaptativos LMS (Least Mean Squares)

Por ejemplo en el Artículo, “Logarithmic quantization in the least mean squares algorithm” de Digital Signal Processing, se destaca que las técnicas del filtrado adaptativo han ido ganando la atención en aplicaciones industriales; siendo algunos ejemplos de estas aplicaciones las estimaciones de canales y ecualización en comunicaciones de sistemas, cancelación de eco y sistemas de identificación. El algoritmo LMS es muy utilizado y bien establecido en algoritmos de filtrado adaptativo. Esto es usado en estimación de parámetros o tamaños del proceso de medida.

Otro ejemplo sobre aplicaciones del filtrado LMS, lo podemos encontrar en el artículo “Block LMS - based source localization using range measurement” de Digital Signal Processing, nos relata también que uno de los usos del filtrado adaptativo LMS se encuentra en la estimación de canal y la reducción de ruido. Además cabe resaltar, que en el artículo hacen mención a que el filtro adaptativo LMS presenta un número de ventajas frente a sus semejantes filtros adaptativos, debido a su robusto comportamiento cuando implementa en precisiones finitas de hardware, y simplicidad computacional para muchas situaciones.

En el artículo “A frequency domain LMS algorithm with dynamic selection of frequency bins” de Circuits System and Signal Processing, mencionan que el filtrado adaptativo es una importante técnica en un rango amplio de aplicaciones como son la ecualización adaptativa, la cancelación de error e identificación de sistemas. Además, también mencionan que de entre todos los filtros adaptativos, el algoritmo del LMS es el más popular y el de mayor adopción por su simplicidad y baja robustez.

Sobre el tema del filtrado adaptativo, a nivel general, en toda su gama de distintas clases que presenta, encontramos sendos artículos en los que se destaca algunas de sus aplicaciones a nivel industrial y destacan además un poco el funcionamiento de los mismos.

Por ejemplo si observamos el artículo “Algorithmic Trading using phase synchronization” de IEEE Journal of Selected Topics in Signal Processing, nos comentan de los filtros adaptativos MLMS Multichannel –Least mean Square, escogido por las siguientes razones, es un algoritmo estocástico que puede operar en altos entornos no estacionarios en tiempo real y sus cuentas de naturaleza multifactorial para el acoplamiento inter-canal. Los algoritmos MLMS es empleada en configuración de predicción en orden de extrapolar el filtrado de señales.

En el artículo “Complex Adaptive LMS Algorithm employing the conjugate gradient principle” de Circuits Systems and Signal Processing, hacen referencia a algoritmos CBC-LMS, Complex Block LMS. El comportamiento de la técnica CBC-LMS es testeado en aplicaciones de estimaciones de canales inalámbricos y ecualización. Además también se hace mención de los filtros adaptativos OBA-LMS, Complex Optimal Block LMS, muy parecido al CBC-LMS. Según el artículo mencionado, deducen que la técnica del CBC-LMS converge más rápido con precisión comparable y menor complejidad computacional.

El artículo “Dual-Antena RF CMOS Front-End for Interferer Removal in Ultra-Wideband Systems”, de Circuits Systems and Signal Processing, comentan del uso de los filtros LMS como un enfoque de adaptación para la eliminación de interferencias para ser usado en distintas aplicaciones.

El artículo “FPGA-Implementation of parallel and sequential architectures for adaptive noise”, de Circuits Systems and Signal Processing, nos hablan de filtros DLMS, Delayed LMS; quienes se caracterizan por inserter un “delay” en el bucle de realimentación de error del algoritmo LMS; y su aplicación con propósitos de y aplicaciones para quitar las interferencias provenientes de señales ECG (Electrocardiography) y ruido blanco de señales de voz.

El artículo “Block filtered-s least mean square algorithm for active control of no-linear noise systems”, de IET Signal Processing, relata que en algunas aplicaciones como las de los auriculares, el conducto del sistema de aire acondicionado, etc, necesitan reducir el nivel de ruido no deseado. Para solucionar esto, se utiliza el FXLMS o que en sus siglas en ingles se denomina Filtered X least mean square, quien es muy usado para ANC (Active Noise Control) de procesos de ruido lineal, siendo ineficiente en la supresión de procesos de ruido no lineal.

9. CONCLUSIONES

Este proyecto está comprendido en tres áreas para su realización y ejecución, que fueron ejecutadas desde el inicio hasta el último día de trabajo.

En la primera área de aplicación, diseñamos filtros paso bajo, paso alto, paso banda y paso banda eliminada, en lo que son los filtros de butterworth, filtros Chebyshev, de tipo uno como de tipo dos y filtros elípticos.

Con esta primera parte, lo que se quiere es conocer, o en nuestro caso, recordar el entorno de Matlab, en sus distintas ecuaciones prediseñadas que nos ofrece el mencionado entorno, como también nos permite conocer un poco las características de estos filtros. Para posteriormente probar dichos filtros en el DSP.

En la segunda etapa, y tras recordar un poco el entorno de Matlab, nos centramos en la elaboración y/o diseño de nuestro filtro adaptativo LMS; experimentado primero con Matlab, para como ya se dijo, entender y comprender el comportamiento del mismo. Cuando ya teníamos claro esta parte, procedimos a “cargar” el código en el DSP, compilarlo y depurarlo, realizando estas últimas acciones gracias al Visual DSP++, que nos resulto un tanto familiar, puesto que se parece bastante a un software usado durante la carrera, el Keil uVision2.

Resaltaremos que durante esta segunda etapa se empezó a excitar las entradas del sistema, con señales provenientes del Cool Edit Pro, y además para saber cómo se comportaba el filtro adaptativo LMS, se utilizó señales provenientes de un generador de funciones, para obtener de esta manera un desfase entre las dos señales de entrada; aunque también se utilizó el propio Cool Edit Pro para obtener señales desfasadas, pero debido que la fase tres no podíamos usar el mencionado software, realizamos pruebas con el generador de funciones.

Finalmente, en la tercera etapa, y tras comprobar el funcionamiento deseado de nuestro filtro adaptativo DSP con señales de entrada simuladas, pasamos a un laboratorio, en donde se utilizó señales provenientes del acelerómetro 4000A, y por supuesto, del generador de funciones; que como se explico en paginas anteriores, sirvió para la formación de nuestra señal de referencia, que permitirá la eliminación de una de las frecuencias que se emitirá del acelerómetro.

Por último, cabe resaltar que pudimos obtener un comportamiento del filtro adaptativo LMS adecuado, y como se esperaba. Realizamos pruebas, como se comentó, con señales de entrada desfasadas, y obtuvimos curiosas respuestas a la salida del sistema, como son que la frecuencia a eliminar, mientras más desfasado estén estas señales, mas se notaba. Solucionando este punto al aumentar el orden del filtro. Es muy importante también comentar la elección que tuvimos del "tamaño de paso", o "μ" utilizado, el cual no puede ser ni muy alto ni muy bajo, ya que si es muy grande la convergencia es rápida y errática, caso contrario si el tamaño de paso es muy bajo, la convergencia es lenta y suave.

Finalmente podemos concluir que pese a que los filtros digitales probados en la primera etapa son útiles, para tener una respuesta lo más ideal posible hay que tener en cuenta el orden del filtro, el cual debe ser muy alto para que las frecuencias próximas a la frecuencia de corte, no se atenúen. En cambio, en los filtros adaptativos LMS, si queremos por ejemplo, eliminar una señal de entre tres señales, como se mostró en imágenes y en apartados anteriores, sólo basta con introducir la frecuencia a eliminar, por una de las entradas del filtro, en concreto la señal de referencia. De esta manera y como bien se ha demostrado, podemos eliminar una señal de entre estas tres, de manera que las otras dos, no se vean afectadas por el procedimiento.

10. BIBLIOGRAFÍA

Autores: Hernández, W., Vicente, J., Sergiyenko, O., Fernández, E.

Título: Improving the Response of Accelerometers for Automotive Applications by Using LMS Adaptive Filters

Autor: Juan Jesús Tortajada Cordero

Título: Sistemas multimedia – Cool Edit Pro

Autor: I. Santamaría

Título: Tema 6, Filtrado Óptimo y Filtrado Adaptativo

http://www.gtas.dicom.unican.es/files/docencia/tds/TEMA6wp.pdf

Autor: José Julio Hernández Fernández

Título: Filtros Adaptativos

http://bibing.us.es/proyectos/abreproy/11284/fichero/Volumen+1%252FCap%EDtulo+2.pdf

Autor: Universidad Nacional de Córdoba

Título: Filtros Adaptativos – LMS – RMS – Filtro Kalman

http://www.dsp.efn.unc.edu.ar/documentos/Filtros_adaptivos.pdf

Enlaces:

http://www.analog.com/static/imported-files/software_manuals_legacy/63652593555244616414545245_getstarted_man.pdf

http://www.analog.com/static/imported-files/application_notes/TN-AD1836_21161.pdf

http://pdf1.alldatasheet.es/datasheet-pdf/view/104880/AD/ADSP-21161N.html

http://www.analog.com/static/imported-files/processor_manuals/63163498310846ADSP_21161_HRM_WEB.pdf

http://www.datasheetcatalog.org/datasheet/analogdevices/344740003AD1836_prc.pdf

http://www.analog.com/static/imported-files/processor_manuals/63163498310846ADSP_21161_HRM_WEB.pdf

http://www.mathworks.es/products/matlab/

http://www.slideshare.net/kevinXD123/ss-cap7-diseno-filtros-fir

http://www.analog.com/en/index.html

http://nees.ucsd.edu/facilities/docs/4000a.pdf

ANEXOS

ANEXO A

```
#include "ADDS_21161_EzKit.h"
#include <def21161.h>

#include <signal.h>

float * DelayLine;
int Index  = 0;

float in[1024];
float out[1024];

float muestra_actual=0;
float muestra_1=0;
float muestra_2=0;
float muestra_3=0;
float muestra_4=0;
float muestra_5=0;
float muestra_6=0;
float muestra_7=0;
float muestra_8=0;
float muestra_9=0;
float muestra_10=0;
float muestra_11=0;
float muestra_12=0;
float muestra_13=0;
float muestra_14=0;
float muestra_15=0;
float muestra_16=0;
float muestra_17=0;
float muestra_18=0;
float muestra_19=0;
float muestra_20=0;


float salida=0;
float salida_1=0;
float salida_2=0;
float salida_3=0;
float salida_4=0;
float salida_5=0;
float salida_6=0;
float salida_7=0;
float salida_8=0;
float salida_9=0;
float salida_10=0;
float salida_11=0;
float salida_12=0;
float salida_13=0;
float salida_14=0;
float salida_15=0;
float salida_16=0;
float salida_17=0;
float salida_18=0;
float salida_19=0;
float salida_20=0;


unsigned int i=0, j=0;

void     Process_Samples( int sig_int)
{
         Receive_Samples();
```

```
        /* Perform AD1836/AD1852/SPDIF Audio Processing Here */

    /*buffer de entrada*/
        if (i<1024)
        {
        in[i]=Left_Channel_In0;
            i++;
        }else
                i=0;


        // Tratamiento de señal (filtro)

        muestra_actual=Left_Channel_In0;

                /*PRUEBA FILTRO BUTTER fc=5kHz Paso Bajo
                Right_Channel_Out1= 0.005542*muestra_actual + 0.02217*m1 +
                0.03325*m2 + 0.02217*m3 +0.00542*m4 + 2.302*s1
                -2.209*s2 + 0.9922* s3 - 0.1742*s4;*/


                /*PRUEBA FILTRO BUTTER fc=1kHz Paso Bajo
                Right_Channel_Out1= 0.000247*muestra_actual +
0.000741*muestra_1 + 0.000741*muestra_2 + 0.000247*muestra_3 + 2.738*salida_1
- 2.51*salida_2 + 0.7695*salida_3;*/

                /*PRUEBA FILTRO BUTTER fc=10kHz Paso Bajo
                Right_Channel_Out1= 0.002941*muestra_actual +
0.02353*muestra_1 + 0.08236*muestra_2 + 0.1647*muestra_3 + 0.2059*muestra_4 +
0.1647*muestra_5 +0.08236*muestra_6 + 0.02353*muestra_7 + 0.002941*muestra_8 +
1.325*salida_1 - 1.771*salida_2 + 1.188*salida_3 - 0.6775*salida_4 +
0.2345*salida_5 - 0.06039*salida_6 + 0.00864*salida_7 - 0.0006076*salida_8;*/


                /*PRUEBA FILTRO BUTTER fc=8kHz Paso Bajo orden 3*/
                //Right_Channel_Out1= 0.06386*muestra_actual +
0.1916*muestra_1 + 0.1916*muestra_2 + 0.06386*muestra_3 + 0.9658*salida_1 -
0.5826*salida_2 + 0.106*salida_3;


                /*PRUEBA FILTRO CHEBY fc=1kHz Paso Bajo
                Right_Channel_Out1= 0.0001851*muestra_actual +
0.0005552*muestra_1 + 0.0005552*muestra_2 + 0.0001851*muestra_3 +
2.824*salida_1 - 2.674*salida_2 + 0.8488*salida_3;*/


                /*PRUEBA FILTRO CHEBY1 fc=8kHz Paso Bajo Orden 3*/
                //Right_Channel_Out1= 0.05805*muestra_actual +
0.1741*muestra_1 + 0.1741*muestra_2 + 0.05805*muestra_3 + 1.179*salida_1 -
0.918*salida_2 + 0.2742*salida_3;


                /*PRUEBA FILTRO CHEBY2 fc=8kHz Paso Bajo Orden 3*/
                //Right_Channel_Out1= 0.7777*muestra_actual +
0.1795*muestra_1 + 0.1795*muestra_2 + 0.7777*muestra_3 + 0.07023*salida_1 -
0.3916*salida_2 - 0.5931*salida_3;

                /*PRUEBA FILTRO ELIPT fc=8kHz Paso Bajo    orden 3*/
                //Right_Channel_Out1= 0.1622*muestra_actual +
0.1325*muestra_1 + 0.1325*muestra_2 + 0.1622*muestra_3 + 1.13*salida_1 -
0.9791*salida_2 + 0.2596*salida_3;

        ////////////////////////////////////////

                /*PRUEBA FILTRO BUTTER fc=8kHz Paso Alto Orden 3*/
```

```
                //Right_Channel_Out1= 0.3318*muestra_actual -
0.9954*muestra_1 + 0.9954*muestra_2 - 0.3318*muestra_3 + 0.9658*salida_1 -
0.5826*salida_2 + 0.106*salida_3;

                /*PRUEBA FILTRO BUTTER fc=7kHz Paso Alto Orden 5*/
                //Right_Channel_Out1= 0.212*muestra_actual - 1.06*muestra_1 +
2.12*muestra_2 - 2.12*muestra_3 + 1.06*muestra_4 - 0.212*muestra_5 +
0.106*muestra_6 + 2.059*salida_1 - 2.133*salida_2 + 1.189*salida_3 -
0.3573*salida_4 + 0.04489*salida_5;

                /*PRUEBA FILTRO CHEBY1 fc=8kHz Paso Alto Orden 3*/
                //Right_Channel_Out1= 0.3236*muestra_actual -
0.9707*muestra_1 + 0.9707*muestra_2 - 0.3236*muestra_3 + 0.9215*salida_1 -
0.6422*salida_2 + 0.02473*salida_3;

                /*PRUEBA FILTRO CHEBY2 fc=8kHz Paso Alto Orden 3*/
                  //Right_Channel_Out1= 0.888*muestra_actual -
1.954*muestra_1 + 1.954*muestra_2 - 0.888*muestra_3 + 2.005*salida_1 -
1.889*salida_2 + 0.7884*salida_3;


                /*PRUEBA FILTRO ELIPT fc=8kHz Paso Alto Orden 3*/
                //Right_Channel_Out1= 0.4355*muestra_actual - 1.101*muestra_1
+ 1.101*muestra_2 - 0.4355*muestra_3 + 1.1*salida_1 - 0.8621*salida_2 +
0.1119*salida_3;



        ////////////////////////////////
                /*PRUEBA FILTRO BUTTER fcmin=5kHz fcmax=12kHz
                  Paso banda Orden 8*/
                //Right_Channel_Out1= 0.001664*muestra_actual -
0.01331*muestra_2 + 0.04658*muestra_4 - 0.09316*muestra_6  + 0.1164*muestra_8
- 0.09316*muestra_10 + 0.04658*muestra_12 - 0.01331*muestra_14 +
0.001664*muestra_16 + 4.651*salida_1 - 10.98*salida_2 + 17.46*salida_3 -
21*salida_4 + 20.07*salida_5 - 15.55*salida_6 + 9.863*salida_7 -
5.156*salida_8 + 2.223*salida_9 - 0.781*salida_10 + 0.2194*salida_11 -
0.04884*salida_12 + 0.008403*salida_13 - 0.0009754*salida_14;

                /*PRUEBA FILTRO CHEBY1 fcmin=5kHz fcmax=12kHz
                  Paso banda Orden 5*/
                //Right_Channel_Out1= 0.007401*muestra_actual -
0.037*muestra_2 + 0.07401*muestra_4 - 0.07401*muestra_6 + 0.037*muestra_8 -
0.007401*muestra_10 + 3.391*salida_1 - 6.573*salida_2 + 8.787*salida_3 -
8.981*salida_4 + 7.006*salida_5 - 4.181*salida_6 + 1.813*salida_7 -
0.5361*salida_8 + 0.0862*salida_9 - 0.002298*salida_10;

                /*PRUEBA FILTRO CHEBY2 fcmin=5kHz fcmax=12kHz
                  Paso banda Orden 5*/
                //Right_Channel_Out1= 0.7821*muestra_actual - 1.874*muestra_1
+ 1.854*muestra_2 - 1.498*muestra_3 + 1.301*muestra_4 - 1.301*muestra_6 +
1.498*muestra_7 - 1.854*muestra_8 + 1.874*muestra_9 - 0.7821*muestra_10 +
2.481*salida_1 - 2.686*salida_2 + 2.283*salida_3 - 1.949*salida_4 +
0.5478*salida_5 + 0.7821*salida_6 - 0.9403*salida_7 + 1.256*salida_8 -
1.413*salida_9 + 0.6117*salida_10;

                /*PRUEBA FILTRO ELLIP fcmin=5kHz fcmax=12kHz
                  Paso banda Orden 5*/
                //Right_Channel_Out1= 0.1205*muestra_actual -
0.3292*muestra_1 + 0.4495*muestra_2 - 0.4617*muestra_3 + 0.3496*muestra_4 -
0.3496*muestra_6 + 0.4617*muestra_7 - 0.4495*muestra_8 + 0.3292*muestra_9 -
0.1205*muestra_10 + 3.531*salida_1 - 7.328*salida_2 + 10.72*salida_3 -
12.21*salida_4 + 10.72*salida_5 - 7.386*salida_6 + 3.873*salida_7 -
1.483*salida_8 + 0.3264*salida_9 - 0.02758*salida_10;
```

```
                    ////////////////////////////////
                          /* */
                /*PRUEBA FILTRO BUTTER fcmin=5kHz fcmax=12kHz
                  Paso banda Eliminada Orden 5*/
                  //Right_Channel_Out1= 0.212*muestra_actual -
1.045*muestra_1 + 3.122*muestra_2 - 6.215*muestra_3 + 9.308*muestra_4 -
10.54*muestra_5 + 9.308*muestra_6 - 6.215*muestra_7 + 3.122*muestra_8 -
1.045*muestra_9 + 0.212*muestra_10 + 3.481*salida_1 - 7.012*salida_2 +
9.756*salida_3 - 10.44*salida_4 + 8.662*salida_5 - 5.684*salida_6 +
2.872*salida_7 - 1.103*salida_8 + 0.2869*salida_9 - 0.04489*salida_10;

                /*PRUEBA FILTRO CHEBY1 fcmin=5kHz fcmax=12kHz
                  Paso banda Eliminada Orden 6*/
                  //Right_Channel_Out1= 0.09316*muestra_actual -
0.5513*muestra_1 + 1.918*muestra_2 - 4.544*muestra_3 + 8.157*muestra_4 -
11.4*muestra_5 + 12.75*muestra_6 - 11.4*muestra_7 + 8.157*muestra_8 -
4.544*muestra_9 + 1.918*muestra_10 - 0.5513*muestra_11 + 0.09316*muestra_12 +
3.736*salida_1 - 7.617*salida_2 + 11.18*salida_3 - 13.1*salida_4 +
12.25*salida_5 - 9.309*salida_6 + 5.992*salida_7 - 3.293*salida_8 +
1.487*salida_9 - 0.6438*salida_10 + 0.3015*salida_11 - 0.08841*salida_12;

                /*PRUEBA FILTRO CHEBY2 fcmin=5kHz fcmax=12kHz
                  Paso banda Eliminada Orden 4*/
                  //Right_Channel_Out1= 0.9028*muestra_actual -
3.194*muestra_1 + 7.251*muestra_2 - 10.97*muestra_3 + 12.78*muestra_4 -
10.97*muestra_5 + 7.251*muestra_6 - 3.194*muestra_7 + 0.9028*muestra_8 +
3.452*salida_1 - 7.621*salida_2 + 11.23*salida_3 - 12.77*salida_4 +
10.68*salida_5 - 6.887*salida_6 + 2.955*salida_7 - 0.8151*salida_8;

                /*PRUEBA FILTRO ELLIP fcmin=5kHz fcmax=12kHz
                  Paso banda Eliminada Orden 4*/
                  //Right_Channel_Out1= 0.4351*muestra_actual -
1.548*muestra_1 + 3.533*muestra_2 - 5.37*muestra_3 + 6.27*muestra_4 -
5.37*muestra_5 + 3.533*muestra_6 - 1.548*muestra_7 + 0.4351*muestra_8 +
2.866*salida_1 - 5.066*salida_2 + 6.315*salida_3 - 6.205*salida_4 +
4.461*salida_5 - 2.498*salida_6 + 1.013*salida_7 - 0.2803*salida_8;

                muestra_20=muestra_19;
                muestra_19=muestra_18;
                muestra_18=muestra_17;
                muestra_17=muestra_16;
                muestra_16=muestra_15;
                muestra_15=muestra_14;
                muestra_14=muestra_13;
                muestra_13=muestra_12;
                muestra_12=muestra_11;
                muestra_11=muestra_10;
                muestra_10=muestra_9;
                muestra_9=muestra_8;
                muestra_8=muestra_7;
                muestra_7=muestra_6;
                muestra_6=muestra_5;
                muestra_5=muestra_4;
                muestra_4=muestra_3;
                muestra_3=muestra_2;
                muestra_2=muestra_1;
                muestra_1=muestra_actual;

                salida_20=salida_19;
                salida_19=salida_18;
                salida_18=salida_17;
                salida_17=salida_16;
                salida_16=salida_15;
                salida_15=salida_14;
                salida_14=salida_13;
                salida_13=salida_12;
```

```
                salida_12=salida_11;
                salida_11=salida_10;
                salida_10=salida_9;
                salida_9=salida_8;
                salida_8=salida_7;
                salida_7=salida_6;
                salida_6=salida_5;
                salida_5=salida_4;
                salida_4=salida_3;
                salida_3=salida_2;
                salida_2=salida_1;
                salida_1=Right_Channel_Out1;

    /*buffer de salida*/

        if (j<1024)
        {
            out[j]=Right_Channel_Out1;
            j++;
        }else
                j=0;

        Transmit_Samples();
}


void main()
{

        /* Setup Interrupt edges and flag I/O directions */
        Setup_ADSP21161N();

        /* Setup SDRAM Controller */
        Setup_SDRAM();

        Setup_AD1836();
        Init_AD1852_DACs();

        Program_SPORT02_TDM_Registers();
        Program_SPORT02_DMA_Channels();

        interruptf(     SIG_SP0I,       Process_Samples);

        *(int *) SP02MCTL |= MCE;

        DelayLine = (float *) 0x00200000;

        for (;;)
                asm("idle;");


}
```

ANEXO B

```
#include "ADDS_21161_EzKit.h"
#include <def21161.h>

#include <signal.h>
#include "math.h"
#include "stdio.h"

float * DelayLine;
int Index  = 0;

#define N 1024      //tamaño de los buffers
#define orden 200   //orden del filtro
#define mu 0.0001   //tamaño del paso

/* Variables e inicializaciones señales LMS */

float error=0;     //señal de error (salida real del sistema)
float entrada=0;   //señal entrada

float x[orden];    //señal de referencia (señal a eliminar)
float salida;      //señal de salida,aproximacion señal referencia

/* Vector de coeficientes del filtro */

float w[orden];

/* Variables para recorrer los buffers entrada y salida */

unsigned int n=0;


void     Process_Samples( int sig_int)
{
         Receive_Samples();

                                                  /* Algoritmo LMS */
         //Exitacion de señales de entrada y referencia

         entrada     = Left_Channel_In0;
         x[0]        = Left_Channel_In1;

         //y=SUM(Wk*x[n-k]); desde k=0 : M-1
         /*Obtenemos la salida, que será una
       aproximación de la señal de referencia*/
         for (n=0;n<orden;n++)
         {
        salida=w[n]*x[n+1]+salida;
         }

         //e=x-y
         /*Obtenemos la señal de salida real del sistema,
      la cual será la diferencia entre la señal de entrada
      con la señal aproximada a la señal de referencia*/
    error = entrada - salida;

    //ecuacion LMS: h[n+1]=h[n]+mu*d[n]*e[n]
         /*Algoritmo del LMS*/
    for (n=0;n<orden;n++)
         w[n] = w[n] + mu*x[n+1]*error;

    /*obtenemos señal deseada en n-1*/

    for (n=orden;n>0;n--)
```

```
        x[n]=x[n-1];

    //sacamos la señal de salida

    Right_Channel_Out1 = error;

}


void main()
{

        /* Setup Interrupt edges and flag I/O directions */
        Setup_ADSP21161N();

        /* Setup SDRAM Controller */
        Setup_SDRAM();

        Setup_AD1836();
        Init_AD1852_DACs();

        Program_SPORT02_TDM_Registers();
        Program_SPORT02_DMA_Channels();


        interruptf(     SIG_SP0I,       Process_Samples);

        *(int *) SP02MCTL |= MCE;

        DelayLine = (float *) 0x00200000;

        for (;;)
                asm("idle;");

}
```